Journal of Machine Learning Research XX (2013) 1-61 Submitted 2/13; Published XX/XX

Reasoning about Independence in Probabilistic Models of Relational Data

Marc Maier MAIER@CS.UMASS.EDU

Katerina Marazopoulou KMARAZO@CS.UMASS.EDU

David Jensen JENSEN@CS.UMASS.EDU
School of Computer Science
University of Massachusetts Amherst
Amherst, MA 01003, USA

Editor:

Abstract

We extend the theory of d-separation to cases in which data instances are not independent and identically distributed. We show that applying the rules of d-separation directly to the structure of probabilistic models of relational data inaccurately infers conditional independence. We introduce *relational d-separation*, a theory for deriving conditional independence facts from relational models. We provide a new representation, the *abstract ground graph*, that enables a sound, complete, and computationally efficient method for answering d-separation queries about relational models, and we present empirical results that demonstrate effectiveness.

Keywords: relational models, d-separation, conditional independence, lifted representations, directed graphical models

1. Introduction

The rules of d-separation can algorithmically derive all conditional independence facts that hold in distributions represented by a Bayesian network. In this paper, we show that d-separation may not correctly infer conditional independence when applied directly to the graphical structure of a relational model. We introduce the notion of *relational d-separation*—a graphical criterion for deriving conditional independence facts from relational models—and define its semantics to be consistent with traditional d-separation (i.e., it claims independence only when it is guaranteed to hold for all model instantiations). We present an alternative, lifted representation—the *abstract ground graph*—that enables an algorithm for deriving conditional independence facts from relational models. We show that this approach is sound, complete, and computationally efficient, and we provide an empirical demonstration of effectiveness across synthetic causal structures of relational domains.

The main contributions of this work are:

- A precise formalization of fundamental concepts of relational data and relational models necessary to reason about conditional independence (Section 4)

- A formal definition of *d*-separation for relational models analogous to *d*-separation for Bayesian networks (Section 5)
- The abstract ground graph—a lifted representation that *abstracts* all possible ground graphs of a given relational model structure, as well as proofs of the soundness and completeness of this abstraction (Section 5.1)
- Proofs of soundness and completeness for a method that answers relational *d*-separation queries (Section 5.2)

We also provide an empirical comparison of relational *d*-separation to traditional *d*-separation applied directly to relational model structure, showing that, not only would most queries be undefined, but those that can be represented yield an incorrect judgment of conditional independence up to 50% of the time (Section 6). Finally, we offer additional empirical results on synthetic data that demonstrate the effectiveness of relational *d*-separation with respect to complexity and consistency (Section 7). The remainder of this introductory section first gives a brief overview of Bayesian networks and their generalization to relational models and then describes why *d*-separation is a useful theory.

1.1 From Bayesian Networks to Relational Models

Bayesian networks are a widely used class of graphical models that are capable of compactly representing a joint probability distribution over a set of variables. The joint distribution can be factored into a product of conditional distributions by assuming that variables are independent of their non-descendants given their parents (the Markov condition). The Markov condition ties the structure of the model to the set of conditional independencies that hold over all probability distributions the model can represent. Accurate reasoning about such conditional independence facts is the basis for constraint-based algorithms, such as PC and FCI (Spirtes et al., 2000), and hybrid approaches, such as MMHC (Tsamardinos et al., 2006), that are commonly used to learn the structure of Bayesian networks. Under a small number of assumptions and with knowledge of the conditional independencies, these algorithms can identify causal structure (Pearl, 2000; Spirtes et al., 2000).

Deriving the full set of conditional independencies implied by the Markov condition is complex, requiring manipulation of the joint distribution and various probability axioms. Fortunately, the exact same set of conditional independencies entailed by the Markov condition are also entailed by *d*-separation, a set of graphical rules that algorithmically derive conditional independence facts directly from the graphical structure of the model. That is, the Markov condition and *d*-separation are equivalent approaches for producing conditional independence from Bayesian networks (Verma and Pearl, 1988; Geiger and Pearl, 1988; Neapolitan, 2004). When interpreting a Bayesian network causally, the causal Markov condition (variables are independent of their non-effects given their direct causes) and *d*-separation have been shown to provide the correct connection between causal structure and conditional independence (Scheines, 1997).

Bayesian networks assume that data instances are independent and identically distributed, but many real-world systems are characterized by interacting heterogeneous entities. For example, social network data consist of individuals, groups, and their relationships; citation data involve researchers collaborating and authoring scholarly papers that cite prior work; and sports data include players, coaches, teams, referees, and their competitive interactions. Over the past 15 years, researchers in statistics and computer science have devised more expressive classes of directed graphical models, such as probabilistic relational models (PRMs), which remove the assumptions of independent and identically distributed instances to more accurately describe these types of domains (Getoor and Taskar, 2007). Relational models generalize other classes of models that incorporate interference, spillover effects, or violations of the stable unit treatment value assumption (SUTVA) (Hudgens and Halloran, 2008; Tchetgen Tchetgen and VanderWeele, 2012) and multilevel or hierarchical models (Gelman and Hill, 2007).

Many practical applications have also benefited from learning and reasoning with relational models. Examples include analysis of gene regulatory interactions (Segal et al., 2001), scholarly citations (Taskar et al., 2001), ecosystems (D'Ambrosio et al., 2003), biological cellular networks (Friedman, 2004), epidemiology (Getoor et al., 2004), and security in information systems (Sommestad et al., 2010). The structure and parameters of these models can be learned from a relational data set. The model is typically used either to predict values of certain attributes (e.g., topics of papers) or the structure is examined directly (e.g., to determine predictors of disease spread). A major goal in many of these applications is to promote understanding of a domain or to determine causes of various outcomes. However, as with Bayesian networks, to effectively interpret and reason about relational models causally, it is necessary to understand their conditional independence implications.

1.2 Why *d*-Separation Is Useful

A Bayesian network, as a model of a joint probability distribution, enables a wide array of useful tasks by supporting inference over an entire set of variables. Bayesian networks have been successfully applied to model many domains, ranging from bioinformatics and medicine to computer vision and information retrieval. Naïvely specifying a joint distribution by hand requires an exponential number of states; however, Bayesian networks leverage the Markov condition to factor a joint probability distribution into a compact product of conditional probability distributions.

The theory of *d*-separation is an alternative to the Markov condition that provides equivalent implications. It provides an algorithmic framework for deriving the conditional independencies encoded by the model. These conditional independence facts are guaranteed to hold in every joint distribution the model represents and, consequently, in any data instance sampled from those distributions. The semantics of holding across all distributions is the main reason why *d*-separation is useful, enabling two large classes of applications:

(1) *Identification of causal effects*: The theory of *d*-separation connects the causal structure encoded by a Bayesian network to the set of probability distributions it can represent. On this basis, many researchers have developed accompanying theory that describes the conditions under which certain causal effects are identifiable (uniquely known) and algorithms for deriving those quantities from the joint distribution. This work enables sound and complete identification of causal effects, not only with respect to conditioning, but also under counterfactuals and interventions—via the *do*-calculus introduced by Pearl (2000)—and in the presence of latent variables (Tian and Pearl, 2002; Huang and Valtorta, 2006; Shpitser and Pearl, 2008).

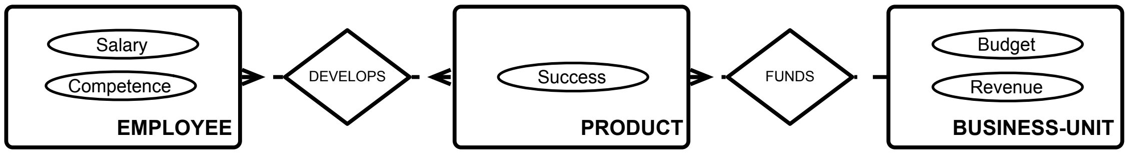
(2) *Constraint-based causal discovery algorithms*: Causal discovery, the task of learning generative models of observational data, superficially appears to be a futile endeavor. Yet learning and reasoning about the causal structure that underlies real domains is a primary goal for many researchers. Fortunately, d-separation offers a connection between causal structure and conditional independence. The theory of d-separation can be leveraged to constrain the hypothesis space by eliminating models that are inconsistent with observed conditional independence facts. While many distributions do not lead to uniquely identifiable models, this approach (under simple assumptions) frequently discovers useful causal knowledge for domains that can be represented as a Bayesian network. This approach to learning causal structure is referred to as the *constraint-based* paradigm, and many algorithms that follow this approach have been developed over the past 20 years, including Inductive Causation (IC) (Pearl and Verma, 1991), PC (Spirtes et al., 2000) and its variants, Three Phase Dependency Analysis (TPDA) (Cheng et al., 1997), Grow-Shrink (Margaritis and Thrun, 1999), Total Conditioning (TC) (Pellet and Elisseeff, 2008), Recursive Autonomy Identification (RAI) (Yehezkel and Lerner, 2009), and hybrid methods that partially employ this approach, including Max-Min Hill Climbing (MMHC) (Tsamardinos et al., 2006) and Hybrid HPC (H2PC) (Gasse et al., 2012).

As described above, relational models more closely represent the real-world domains that many social scientists and other researchers investigate. To successfully learn causal models from observational data of relational domains, we need a theory for deriving conditional independence from relational models. In this paper, we formalize the theory of *relational d-separation* and provide a method for deriving conditional independence facts from the structure of a relational model. In another paper, we have used these results to provide a theoretical framework for a sound and complete constraint-based algorithm—the Relational Causal Discovery (RCD) algorithm (Maier et al., 2013)—that learns causal models of relational domains.

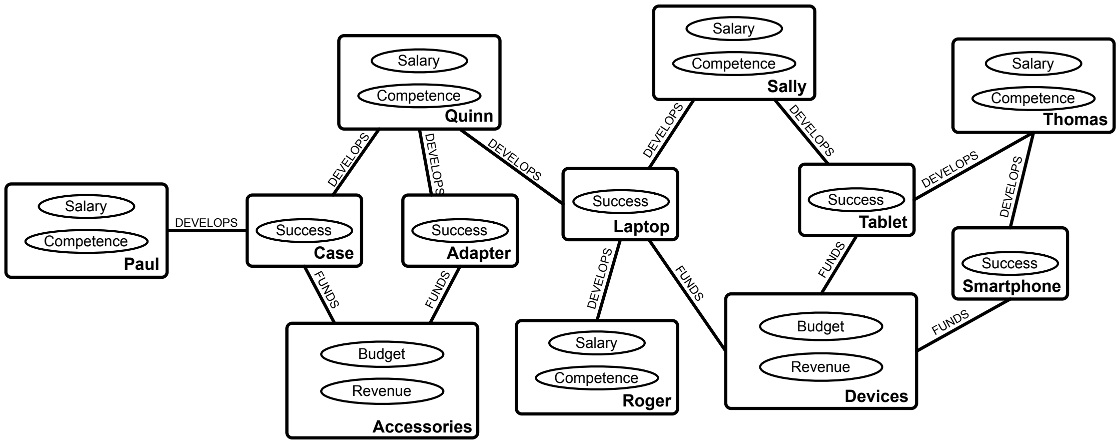
2. Example

Consider a corporate analyst who was hired to identify which employees are effective and productive for some organization. If the company is structured as a pure project-based organization (for which company personnel are structured around projects, not departments), the analyst may collect data as described by the relational schema in Figure 2.1(a). The schema denotes that employees can collaborate and work on multiple products, each of which is funded by a specific business unit. The analyst has also obtained attributes on each entity—salary and competence of employees, the success of each product, and the budget and revenue of business units. In this example, the organization consists of five employees, five products, and two business units, which are shown in the relational skeleton in Figure 2.1(b).

Assume that the organization operates under the model depicted in Figure 2.2(a). For example, the success of a product depends on the competence of employees that develop it, and the revenue of a business unit is influenced by the success of products that it funds. If this were known by the analyst (who happens to have experience in graphical models), then it would be conceivable to spot-check the model and test whether some of the conditional independencies encoded by the model are reflected in the data. The analyst then naïvely



(a) Example relational schema for an organization consisting of employees working on products, which are funded by specific business units within a corporation.



(b) Example fragment of a relational skeleton. Roger and Sally are employees, both of whom develop the Laptop product, but, of the two, only Sally works on product Tablet. Both products Laptop and Tablet are funded by business unit Devices. For convenience, we depict attribute placeholders on each entity instance.

Figure 2.1: An example relational schema and skeleton for the organization domain.

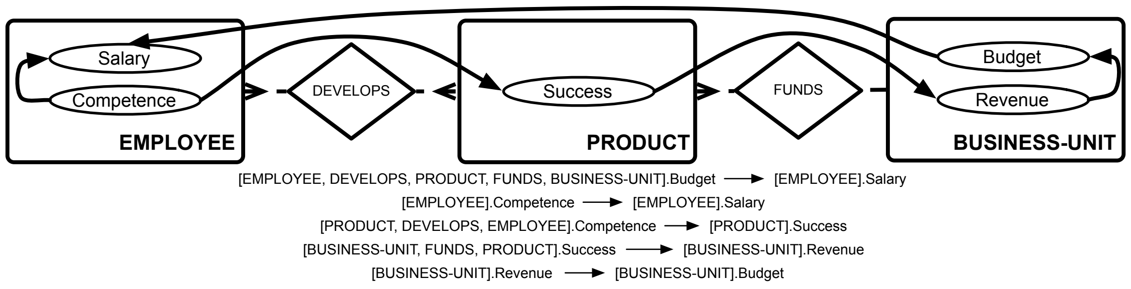
applies d-separation to the model structure in an attempt to derive conditional independencies to test. However, applying d-separation directly to the structure of relational models may not correctly derive conditional independencies, violating the Markov condition. If the analyst were to discover significant and substantive effects, he may believe the model structure is incorrect and needlessly search for alternative dependencies.

Naïvely applying d-separation to the model in Figure 2.2(a) suggests that employee competence is conditionally independent of the revenue of business units given the success of products:

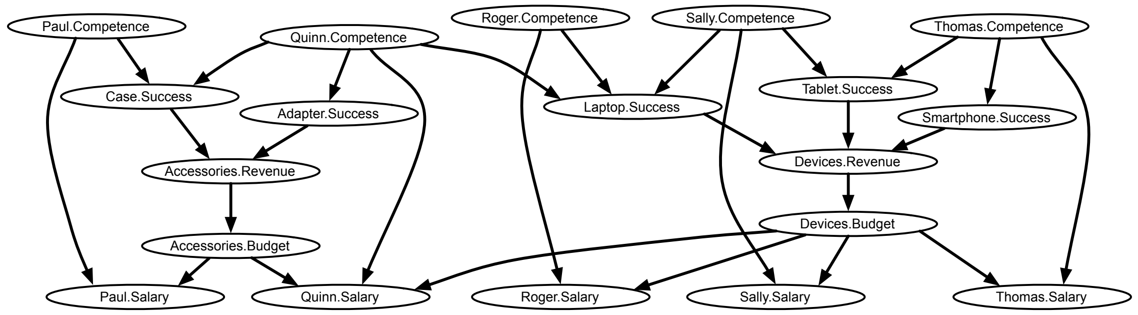
$$\textsc{Employee}.\mathit{Competence} \perp\!\!\!\perp \textsc{Business-Unit}.\mathit{Revenue} \mid \textsc{Product}.\text{Success}$$

To see why this approach is flawed, we must consider the *ground graph*. A necessary precondition for inference is to apply a model to a data instantiation, which yields a ground graph to which d-separation can be applied. For a Bayesian network, a ground graph consists of replicates of the model structure for each data instance. In contrast, a relational model defines a template for how dependencies apply to a data instantiation, resulting in a ground graph with varying structure. See Section 4 for more details on ground graphs.

Figure 2.2(b) shows the ground graph for the relational model in Figure 2.2(a) applied to the relational skeleton in Figure 2.1(b). This ground graph illustrates that, for a single employee, simply conditioning on the success of developed products can activate a path through the competence of other employees who develop the same products—we call


Salary
Competence
EMPLOYEE
DEVELOPS
Success
PRODUCT
FUNDS
Budget
Revenue
BUSINESS-UNIT
[EMPLOYEE, DEVELOPS, PRODUCT, FUNDS, BUSINESS-UNIT].Budget → [EMPLOYEE].Salary
[EMPLOYEE].Competence → [EMPLOYEE].Salary
[PRODUCT, DEVELOPS, EMPLOYEE].Competence → [PRODUCT].Success
[BUSINESS-UNIT, FUNDS, PRODUCT].Success → [BUSINESS-UNIT].Revenue
[BUSINESS-UNIT].Revenue → [BUSINESS-UNIT].Budget

(a) Example relational model. Competence of employees cause the success of products they develop, which in turn influences the revenue received by the business unit funding the product. Additional dependencies involve the budget of business units and employee salaries. The dependencies are specified by relational paths, listed below the graphical model.



(b) Example fragment of a ground graph. The success of product Laptop is influenced by the competence of both Roger and Sally. The revenue of business unit Devices is caused by the success of all its funded products—Laptop, Tablet, and Smartphone.

Figure 2.2: An example relational model and ground graph for the organization domain.

this a *relationally d-connecting path*.[1] Checking d-separation on the ground graph indicates that to d-separate an employee's competence from the revenue of funding business units, we should not only condition on the success of developed products, but also on the competence of other employees who work on those products (e.g., Roger.$Competence \perp\!\!\!\perp$ Devices.$Revenue \mid \{$Laptop.$Success$, Sally.$Competence\}$).

This example also demonstrates that the Markov condition can be violated when directly applied to the structure of a relational model. In this case, the Markov condition according to the model structure in Figure 2.2(a) implies that $P(Competence, Revenue \mid Success) = P(Competence \mid Success)P(Revenue \mid Success)$, that revenue is independent of its non-descendants (competence) given its parents (success). However, the ground graph shows the opposite, for example, $P($Roger.$Competence$, Devices.$Revenue \mid$ Laptop.$Success) \neq P($Roger.$Competence \mid$Laptop.$Success)$ $P($Devices.$Revenue \mid$ Laptop.$Success)$. In fact, for this model, d-separation produces many other incorrect judgments of conditional independence. Through simulation, we found that only 25% of the pairs of variables can even be

1. The indirect effect attributed to a relationally d-connecting path is often referred to as interference, a spillover effect, or a violation of the stable unit treatment value assumption (SUTVA) because the treatment of one instance (employee competence) affects the outcome of another (the revenue of another employee's business unit).

described by direct inspection of this model structure, and of those (such as the above example), 75% yield an incorrect conclusion. This is a single data point of a larger empirical evaluation presented in Section 6. Those results provide quantitative details of how often to expect traditional d-separation to fail when applied to the structure of relational models.

3. Semantics and Alternatives

The example in Section 2 provides a useful basis to describe the semantics imposed by relational d-separation and the characteristics of our approach. There are two primary concepts:

(1) *All-ground-graphs semantics*: It might appear that, since the standard rules of d-separation apply to Bayesian networks and the ground graphs of relational models are also Bayesian networks, that applying d-separation to relational models is a non-issue. However, applying d-separation to a single ground graph may result in potentially unbounded runtime if the instantiation is large (i.e., since relational databases can be arbitrarily large). Further, and more importantly, the semantics of d-separation require that conditional independencies hold across *all possible* model instantiations. Although d-separation can apply directly to a ground graph, these semantics prohibit reasoning about a single ground graph.

The conditional independence facts derived from d-separation hold for all distributions represented by a Bayesian network. Analogously, the implications of *relational* d-separation should hold for all distributions represented by a relational model. It is simple to show that these implications hold for all ground graphs of a Bayesian network—every ground graph consists of a set of disconnected subgraphs, each of which has a structure that is identical to that of the model. However, the set of distributions represented by a relational model depends on both the relational skeleton (constrained by the schema) and the model parameters. That is, the ground graphs of relational models vary with the structure of the underlying relational skeleton (e.g., different products are developed by varying numbers of employees). As a result, answering relational d-separation queries requires reasoning without respect to ground graphs.

(2) *Perspective-based analysis*: Relational models make explicit one implicit choice underlying nearly any form of data analysis. This choice—what we refer to here as a *perspective*—concerns the selection of a particular unit or subject of analysis. For example, in the social sciences, a commonly used acronym is *UTOS*, for framing an analysis by choosing a unit, treatment, outcome, and setting. Any method, such as Bayesian network modeling, that assumes IID data makes the implicit assumption that the attributes on data instances correspond to attributes of a single unit or perspective. In the example, we targeted a specific conditional independence regarding employee instances (as opposed to products or business units).

The concept of perspectives is not new, but it is central to statistical relational learning because relational data sets may be heterogeneous, involving instances that refer to multiple, distinct perspectives. The inductive logic programming (ILP) community has discussed individual-centered representations (Flach, 1999), and many approaches to propositionalizing relational data have been developed to enforce a *single* perspective in order to rely on existing propositional learning algorithms (Kramer et al., 2001). An alternative strategy is to explicitly acknowledge the presence of multiple perspectives and learn jointly among them. This approach underlies many algorithms that learn the types of probabilistic models of relational data applicable in this work, e.g., learning the structure of probabilistic relational models, relational dependency networks, or parametrized Bayesian networks (Friedman et al., 1999; Neville and Jensen, 2007; Schulte et al., 2012).

Often, data sets are derivative, leading to little or no choice about which perspectives to analyze. However, for relational domains, from which these data sets are derived, it is assumed that there are multiple perspectives, and we can dynamically analyze different perspectives. In the example, we chose the employee perspective, and the analysis focused on the dependence between an employee's competence and the revenue of business units that fund developed products. However, if the question were posed from the perspective of business units, then we could conceivably condition on the success of products for each business unit. In this scenario, reasoning about d-separation at the model level *would* lead to a correct conditional independence statement. Some (though fairly infrequent) d-separation queries produce accurate conditional independence facts when applied to relational model structure (see Section 6). However, the model is often unknown, a perspective may be chosen a priori, and a theory that is occasionally correct is clearly undesirable. Additionally, to support constraint-based learning algorithms, it is important to reason about conditional independence implications from different perspectives.

One plausible alternative approach would be to answer d-separation queries by ignoring perspectives and considering just the attribute classes (i.e., reason about *Competence* and *Revenue* given *Success*). However, it remains to define explicit semantics for grounding and evaluating the query based on the relational skeleton. There are at least three options:

- *Construct three sets of variables, including all instances of competence, revenue, and success variables*: Although the ground graph has the semantics of a Bayesian network, there is only a *single* ground graph—one data sample (Xiang and Neville, 2011). Consequently, this analysis would be statistically meaningless and is the primary reason why relational learning algorithms dynamically *generate* propositional data for each instance of a given perspective.
- *Test the Cartesian product of competence and revenue variables, conditioned on all success variables*: Testing all pairs invariably leads to independence. Moreover, these semantics are incoherent; only reachable pairs of variables should be compared. For propositional data, variable pairs are constructed by choosing attribute values, e.g., height and weight, *within* an individual. The same is true for relational data: Only choose the success of products for employees that actually develop them, following the underlying relational connections.
- *Test relationally connected pairs of competence and revenue variables, conditioned on all success variables*: Again, this appears plausible based on traditional d-separation; every instance in the table conditions on the *same* set of success values. Therefore, this is akin to not conditioning because the conditioning variable is a constant.

We argue that the desired semantics are essentially the explicit semantics of perspective-based queries. Therefore, we advocate perspective-based analysis as the *only* statistically and semantically meaningful approach for relational data and models.

Our approach to answering relational d-separation queries incorporates the two aforementioned semantics. In Section 5, we describe a new, lifted representation—the abstract

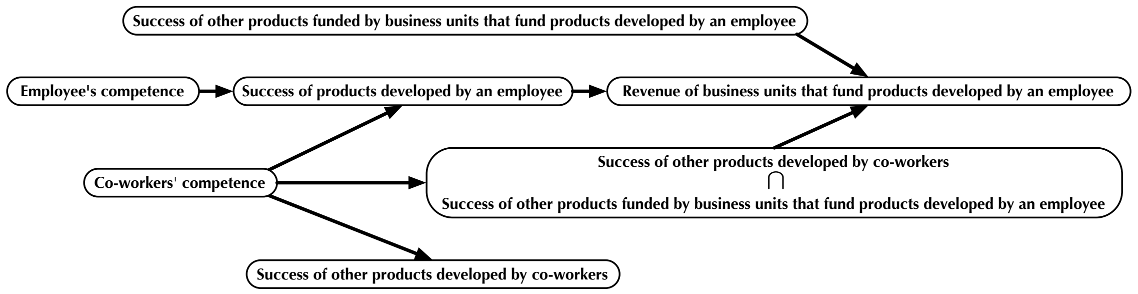

Success of other products funded by business units that fund products developed by an employee
Employee's competence
Success of products developed by an employee
Revenue of business units that fund products developed by an employee
Co-workers' competence
Success of other products developed by co-workers
∩
Success of other products funded by business units that fund products developed by an employee
Success of other products developed by co-workers

Figure 3.1: Example abstract ground graph from the perspective of employees. Nodes are labeled with their intuitive meaning.

ground graph—that is provably sound and complete in its abstraction of all ground graphs for a given relational model. As their name suggests, abstract ground graphs *abstract* all ground graphs of a relational model, representing any potential relationally d-connecting path (recall the example d-connecting path that only manifests in the ground graph). A relational model has a corresponding *set* of abstract ground graphs, one for each perspective (i.e., entity or relationship class in its underlying schema), and can be used to reason about relational d-separation with respect to any given perspective. Figure 3.1 shows a fragment of an abstract ground graph from the employee perspective for the model in Figure 2.2a. The nodes are depicted with their intuitive meaning rather than their actual syntax for this example. Representational details and accompanying theory are presented in Section 5.

4. Concepts of Relational Data and Models

Propositional representations describe domains with a single entity type, but many real-world systems involve multiple types of interacting entities with probabilistic dependencies among their variables. For example, in the model in Figure 2.2(a) the competence of employees affects the success of products they develop. Many researchers have focused on modeling such domains, which are generally characterized as *relational*. These relational representations can be divided into two main categories: *probabilistic graphical models*—such as probablistic relational models (PRMs) (Koller and Pfeffer, 1998), directed acyclic probabilistic entity-relationship (DAPER) models (Heckerman et al., 2004), and relational Markov networks (RMNs) (Taskar et al., 2002)—and *probabilistic logic models*—such as Bayesian logic programs (BLPs) (Kersting and De Raedt, 2002), Markov logic networks (MLNs) (Richardson and Domingos, 2006), parametrized Bayesian networks (PBNs) (Poole, 2003), Bayesian logic (Blog) (Milch et al., 2005), multi-entity Bayesian networks (MEBNs) (Laskey, 2008), and relational probability models (RPMs) (Russell and Norvig, 2010).

To facilitate an extension to the graphical criterion of d-separation, we currently focus on directed, acyclic, graphical models of conditional independence. As most of the above models have similar expressive power, the results in this paper could generalize across representations—even for *undirected* relational models, such as RMNs and MLNs, after moralization. However, we found it simpler to define and prove relevant theoretical properties for relational d-separation in a representation most similar to Bayesian networks. In this section, we formally define the concepts of relational data and models using a similar representation to PRMs and DAPER models.

A relational schema is a top-level description of what data exist in a particular domain. Specifically (adapted from Heckerman et al., 2007):

Definition 4.1 (Relational schema) A *relational schema* $\mathcal{S} = (\mathcal{E}, \mathcal{R}, \mathcal{A}, card)$ consists of a set of entity classes $\mathcal{E} = \{E_1, \ldots, E_m\}$; a set of relationship classes $\mathcal{R} = \{R_1, \ldots, R_n\}$, where each $R_i = \langle E_1^i, \ldots, E_{a_i}^i \rangle$, with $E_j^i \in \mathcal{E}$ and a_i is the arity for R_i; a set of attribute classes $\mathcal{A}(I)$ for each item class $I \in \mathcal{E} \cup \mathcal{R}$; and a cardinality function $card : \mathcal{R} \times \mathcal{E} \rightarrow$ {ONE, MANY}.

A relational schema can be represented graphically with an entity-relationship (ER) diagram. We adopt a slightly modified ER diagram using Barker's notation (1990), where entity classes are rectangular boxes, relationship classes are diamonds with dashed lines connecting their associated entity classes, attribute classes are ovals residing on entity and relationship classes, and cardinalities are represented with crow's foot notation.

Example 4.1 The relational schema $\mathcal{S}$ for the organization domain example depicted in Figure 2.1(a) consists of entities $\mathcal{E}$ = {EMPLOYEE, PRODUCT, BUSINESS-UNIT}; relationships $\mathcal{R}$ = {DEVELOPS, FUNDS}, where DEVELOPS = ⟨EMPLOYEE, PRODUCT⟩, FUNDS = ⟨BUSINESS-UNIT, PRODUCT⟩ and having cardinalities *card*(DEVELOPS, EMPLOYEE) = MANY, *card*(DEVELOPS, PRODUCT) = MANY, *card*(FUNDS, BUSINESS-UNIT) = MANY, and *card*(FUNDS, PRODUCT) = ONE; and attributes $\mathcal{A}$(EMPLOYEE) = {*Competence*, *Salary*}, $\mathcal{A}$(PRODUCT) = {*Success*}, and $\mathcal{A}$(BUSINESS-UNIT) = {*Budget*, *Revenue*}. □

A relational schema is a template for a relational skeleton (also referred to as a data graph by Neville and Jensen, 2007), an instantiation of entity and relationship classes. Specifically (adapted from Heckerman et al., 2007):

Definition 4.2 (Relational skeleton) A *relational skeleton* σ for relational schema $\mathcal{S} = (\mathcal{E}, \mathcal{R}, \mathcal{A}, card)$ specifies a set of entity instances $\sigma(E)$ for each $E \in \mathcal{E}$ and relationship instances $\sigma(R)$ for each $R \in \mathcal{R}$. Relationship instances adhere to the cardinality constraints of $\mathcal{S}$: If $card(R, E)$ = ONE, then for each $e \in \sigma(E)$ there is at most one $r \in \sigma(R)$ such that e participates in r.

For convenience, we use the notation $E \in R$ if entity class E is a component of relationship class R, and, similarly, $e \in r$ if entity instance e is a component of the relationship instance r. We also denote the set of all skeletons for schema $\mathcal{S}$ as $\Sigma_{\mathcal{S}}$.

Example 4.2 The relational skeleton σ for the organization example is depicted in Figure 2.1(b). The sets of entity instances are σ(EMPLOYEE) = {Paul, Quinn, Roger, Sally, Thomas}, σ(PRODUCT) = {Case, Adapter, Laptop, Tablet, Smartphone}, and σ(BUSINESS-UNIT) = {Accessories, Devices}. The sets of relationship instances are σ(DEVELOPS) = {⟨Paul, Case⟩, ⟨Quinn, Case⟩, . . . , ⟨Thomas, Smartphone⟩} and σ(FUNDS) = {⟨Accessories, Case⟩, ⟨Accessories, Adapter⟩, . . . , ⟨Devices, Smartphone⟩}. The relationship instances adhere to their cardinality constraints (e.g., FUNDS is a ONE-to-MANY relationship—within σ(FUNDS), every product has a single business unit, and every business unit may have multiple products). □

In order to specify a model over a relational domain, we must define a space of possible variables and dependencies. Consider the example dependency [Product, Develops, Employee].*Competence* $\rightarrow$ [Product].*Success* from the model in Figure 2.2(a), expressing that the competence of employees developing a product affects the success of that product. For relational data, the variable space includes not only intrinsic entity and relationship attributes (e.g., success of a product), but also the attributes on other entity and relationship classes that are reachable by paths along the relational schema (e.g., the competence of employees that develop a product). We define relational paths to formalize the notion of which item classes are reachable on the schema from a given item class.[2]

Definition 4.3 (Relational path) A *relational path* $[I_j, \ldots, I_k]$ for relational schema $\mathcal{S}$ is an alternating sequence of entity and relationship classes $I_j, \ldots, I_k \in \mathcal{E} \cup \mathcal{R}$ such that:
(1) For every pair of consecutive item classes $[E, R]$ or $[R, E]$ in the path, $E \in R$.
(2) For every triple of consecutive item classes $[E, R, E']$, $E \neq E'$.[3]
(3) For every triple of consecutive item classes $[R, E, R']$, if $R = R'$, then $card(R, E) =$ many.
I_j is called the *base item*, or *perspective*, of the relational path.

Condition (1) enforces that entity classes participate in adjacent relationship classes in the path. Conditions (2) and (3) remove any paths that would invariably reach an empty terminal set (see Definition 4.4 and Appendix C). This definition of relational paths is similar to "meta-paths" and "relevance paths" in similarity search and information retrieval in heterogeneous networks (Sun et al., 2011; Shi et al., 2012). Relational paths also extend the notion of "slot chains" from the PRM framework (Getoor et al., 2007) by including cardinality constraints and formally describing the semantics under which repeated item classes may appear on a path. Relational paths are also a specialization of the first-order constraints on arc classes imposed on DAPER models (Heckerman et al., 2007).

Example 4.3 Consider the example relational schema in Figure 2.1(a). Some example relational paths from the Employee perspective (with an intuitive meaning of what the paths describe) include the following: [Employee] (an employee), [Employee, Develops, Product] (products developed by an employee), [Employee, Develops, Product, Funds, Business-Unit] (business units of the products developed by an employee), and [Employee, Develops, Product, Develops, Employee] (co-workers developing the same products). Invalid relational paths include [Employee, Develops, Employee] (because Employee=Employee and Develops $\in \mathcal{R}$) and [Business-Unit, Funds, Product, Funds, Business-Unit] (because Product $\in \mathcal{E}$ and *card*(Funds, Product) = one). $\square$

Relational paths are defined at the level of relational schemas, and as such are templates for paths in a relational skeleton. An instantiated relational path produces a set of traversals

2. Because the term "path" is also commonly used to describe chains of dependencies in graphical models, we will explicitly qualify each reference to avoid ambiguity.
3. This condition suggests at first glance that self-relationships (e.g., employees manage other employees, individuals in social networks maintain friendships, scholarly articles cite other articles) are prohibited. We discuss this and other model assumptions in Section 8.

on a relational skeleton. However, the quantity of interest is not the traversals, but the set of reachable item instances (i.e., entity or relationship instances). These reachable instances are the fundamental elements that support model instantiations (i.e., ground graphs).

Definition 4.4 (Terminal set) For skeleton $\sigma \in \Sigma_{\mathcal{S}}$ and $i_j \in \sigma(I_j)$, the *terminal set* $P|_{i_j}$ for relational path $P = [I_j, \dots, I_k]$ of length n is defined inductively as

$$P^1|_{i_j} = [I_j]|_{i_j} = \{i_j\}$$

$$\vdots$$

$$P^n|_{i_j} = [I_j, \dots, I_k]|_{i_j} = \bigcup_{i_m \in P^{n-1}|_{i_j}} \Big\{ i_k \mid \big((i_m \in i_k \text{ if } I_k \in \mathcal{R}) \ \vee \ (i_k \in i_m \text{ if } I_k \in \mathcal{E}) \big) \wedge i_k \notin \bigcup_{l=1}^{n-1} P^l|_{i_j} \Big\}$$

A terminal set of a relational path $P = [I_j, \dots, I_k]$ consists of instances of class I_k, the terminal item on the path. Conceptually, a terminal set is produced by traversing a skeleton beginning at a single instance of the base item class, $i_j \in \sigma(I_j)$, following instances of the item classes in the relational path, and reaching a set of instances of class I_k. The term $i_k \notin \bigcup_{l=1}^{n-1} P^l|_{i_j}$ in the definition implies a "bridge burning" semantics under which no item instances are revisited (i_k does not appear in the terminal set of any prefix of P).[4] The notion of terminal sets is a necessary concept for grounding any relational model and has been described in previous work—e.g., for PRMs (Getoor et al., 2007) and MLNs (Richardson and Domingos, 2006)—but has not been explicitly named. We emphasize their importance because terminal sets are also critical for defining relational d-separation, and we formalize the semantics for bridge burning.

Example 4.4 We can generate terminal sets by pairing the set of relational paths for the schema in Figure 2.1(a) with the relational skeleton in Figure 2.1(b). Let Quinn be our base item instance. Then [Employee]$|_{\text{Quinn}}$ = {Quinn}, [Employee, Develops, Product]$|_{\text{Quinn}}$ = {Case, Adapter, Laptop}, [Employee, Develops, Product, Funds, Business-Unit]$|_{\text{Quinn}}$ = {Accessories, Devices}, and [Employee, Develops, Product, Develops, Employee]$|_{\text{Quinn}}$ = {Paul, Roger, Sally}. The bridge burning semantics enforce that Quinn is not also included in this last terminal set. □

For a given base item class, it is common (depending on the schema) for distinct relational paths to reach the same terminal item class. The following lemma states that if two relational paths with the same base item and the same terminal item differ at some point in the path, then for some relational skeleton and some base item instance, their terminal sets will have a non-empty intersection. This property is important to consider for relational d-separation.

Lemma 4.1 *For two relational paths of arbitrary length from I_j to I_k that differ in at least one item class, $P_1 = [I_j, \dots, I_m, \dots, I_k]$ and $P_2 = [I_j, \dots, I_n, \dots, I_k]$ with $I_m \neq I_n$, there exists a skeleton $\sigma \in \Sigma_{\mathcal{S}}$ such that $P_1|_{i_j} \cap P_2|_{i_j} \neq \emptyset$ for some $i_j \in \sigma(I_j)$.*

4. The bridge burning semantics yield terminal sets that are necessarily subsets of terminal sets that would otherwise be produced without bridge burning. Although this appears to be limiting, it actually enables a strictly more expressive class of relational models. See Appendix B for more details and an example.

Proof. See Appendix A.

Example 4.5 Let P_1 = [EMPLOYEE, DEVELOPS, PRODUCT, DEVELOPS, EMPLOYEE, DEVELOPS, PRODUCT], the terminal sets for which yield other products developed by collaborating employees. Let P_2 = [EMPLOYEE, DEVELOPS, PRODUCT, FUNDS, BUSINESS-UNIT, FUNDS, PRODUCT], the terminal sets for which consist of other products funded by the business units funding products developed by a given employee. Intersection among terminal sets for these paths occurs even in the small example skeleton. In fact, the intersection of the terminal sets for P_1 and P_2 is non-empty for all employees. For example, Paul: $P_1|_{\text{Paul}} = \{\text{Adapter, Laptop}\}$ and $P_2|_{\text{Paul}} = \{\text{Adapter}\}$; Quinn: $P_1|_{\text{Quinn}} = \{\text{Tablet}\}$ and $P_2|_{\text{Quinn}} = \{\text{Tablet, Smartphone}\}$. □

Given the definition for relational paths, it is simple to define relational variables and their instances.

Definition 4.5 (Relational variable) A *relational variable* $[I_j, \ldots, I_k].X$ consists of a relational path $[I_j, \ldots, I_k]$ and an attribute class $X \in \mathcal{A}(I_k)$.

As with relational paths, we refer to I_j as the perspective of the relational variable. Relational variables are templates for sets of random variables (see Definition 4.6). Sets of relational variables are the basis of relational d-separation queries, and consequently they are also the nodes of the abstract representation that answers those queries. There is an equivalent formulation in the PRM framework, although not explicitly named (they are simply denoted as attribute classes of **K**-related item classes via slot chain **K**). As they are critical to relational d-separation, we provide this concept with an explicit designation.

Example 4.6 Relational variables for the relational paths in Example 4.3 include intrinsic attributes such as [EMPLOYEE].*Competence* and [EMPLOYEE].*Salary*, and also attributes on related entity classes such as [EMPLOYEE, DEVELOPS, PRODUCT].*Success*, [EMPLOYEE, DEVELOPS, PRODUCT, FUNDS, BUSINESS-UNIT].*Revenue*, and [EMPLOYEE, DEVELOPS, PRODUCT, DEVELOPS, EMPLOYEE].*Salary*. □

Definition 4.6 (Relational variable instance) For skeleton $\sigma \in \Sigma_{\mathcal{S}}$ and $i_j \in \sigma(I_j)$, a *relational variable instance* $[I_j, \ldots, I_k].X|_{i_j}$ for relational variable $[I_j, \ldots, I_k].X$ is the set of random variables $\{i_k.X \mid X \in \mathcal{A}(I_k) \ \wedge \ i_k \in [I_j, \ldots, I_k]|_{i_j} \ \wedge \ i_k \in \sigma(I_k)\}$.

To instantiate a relational variable $[I_j, \ldots, I_k].X$ for a specific base item instance i_j, we first find the terminal set of the underlying relational path $[I_j, \ldots, I_k]|_{i_j}$ and then take the X attributes of the I_k item instances in that terminal set. This produces a set of random variables $i_k.X$, which also correspond to nodes in the ground graph. As a notational convenience, if $\mathbf{X}$ is a set of relational variables, all from a common perspective I_j, then we say that $\mathbf{X}|_{i_j}$ for some item $i_j \in \sigma(I_j)$ is the union of all instantiations, $\{x \mid x \in X|_{i_j} \ \wedge \ X \in \mathbf{X}\}$.

Example 4.7 Instantiating the relational variables from Example 4.6 with base item instance Sally yields [EMPLOYEE].*Competence*$|_{\text{Sally}}$ = {Sally.*Competence*}, [EMPLOYEE, Develops, Product].*Success*$|_{\text{Sally}}$ = {Laptop.*Success*, Tablet.*Success*}, [Employee, Develops, Product, Funds, Business-Unit].*Revenue*$|_{\text{Sally}}$ = {Devices.*Revenue*}, and [Employee, Develops, Product, Develops, Employee].*Salary*$|_{\text{Sally}}$ = {Quinn.*Salary*, Thomas.*Salary*}. □

Given the definitions for relational variables, we can now define relational dependencies.

Definition 4.7 (Relational dependency) A *relational dependency* $[I_j, \ldots, I_k].Y \rightarrow [I_j].X$ is a directed probabilistic dependence from attribute class Y to X through the relational path $[I_j, \ldots, I_k]$.

Depending on the context, $[I_j, \ldots, I_k].Y$ and $[I_j].X$ can be referred to as *treatment* and *outcome*, *cause* and *effect*, or *parent* and *child*. A relational dependency consists of two relational variables having a common perspective. The relational path of the child is restricted to a single item class, ensuring that the terminal sets consist of a single value. This is consistent with PRMs, except that we explicitly delineate dependencies rather than define parent sets of relational variables. Note that relational variables are not nodes in a relational model, but they form the space of parent variables for relational dependencies. The relational path specification (before the attribute class of the parent) is equivalent to a slot chain, as in PRMs, or the logical constraint on a dependency, as in DAPER models.

Example 4.8 The dependencies in the relational model displayed in Figure 2.2(a) can be specified as: [Product, Develops, Employee].*Competence* → [Product].*Success* (product success is influenced by the competence of the employees developing the product), [Employee].*Competence* → [Employee].*Salary* (an employee's competence affects his or her salary), [Business-Unit, Funds, Product].*Success* → [Business-Unit].*Revenue* (the success of the products funded by a business unit influences that unit's revenue), [Employee, Develops, Product, Funds, Business-Unit].*Budget*→[Employee].*Salary* (employee salary is governed by the budget of the business units for which they develop products), and [Business-Unit].*Revenue* → [Business-Unit].*Budget* (the revenue of a business unit influences its budget). □

We now have sufficient information to define relational models.

Definition 4.8 (Relational model) A *relational model* $\mathcal{M}_\Theta$ consists of two parts:

1. The *structure* $\mathcal{M} = (\mathcal{S}, \mathcal{D})$: a schema $\mathcal{S}$ paired with a set of relational dependencies $\mathcal{D}$ defined over $\mathcal{S}$.
2. The *parameters* Θ: a conditional probability distribution $P\big([I_j].X \mid \mathit{parents}([I_j].X)\big)$ for each relational variable of the form $[I_j].X$, where $I_j \in \mathcal{E} \cup \mathcal{R}$, $X \in \mathcal{A}(I_j)$ and $\mathit{parents}\big([I_j].X\big) = \big\{[I_j, \ldots, I_k].Y \mid [I_j, \ldots, I_k].Y \rightarrow [I_j].X \in \mathcal{D}\big\}$ is the set of parent relational variables.

The structure of a relational model can be represented graphically by superimposing dependencies on the ER diagram of a relational schema (see Figure 2.2(a) for an example). A relational dependency of the form $[I_j, \ldots, I_k].Y \rightarrow [I_j].X$ is depicted as a directed arrow from attribute class Y to X with the specification listed separately. Note that the subset of relational variables with singleton paths $[I].X$ in the definition correspond to the set of attribute classes in the schema.

A common technique in relational learning is to use aggregation functions to transform parent multi-sets to single values within the conditional probability distributions. Typically, aggregation functions are simple, such as mean or mode, but they can be complex, such as those based on vector distance or object identifiers, as in the ACORA system (Perlich and Provost, 2006). However, aggregates are a convenience for increasing power and accuracy during learning, but they are not necessary for model specification.

This definition of relational models is consistent with and yields structures expressible as DAPER models (Heckerman et al., 2007). These relational models are also equivalent to PRMs, but we extend slot chains as relational paths and provide a formal semantics for their instantiation. These models are also more general than plate models because dependencies can be specified with arbitrary relational paths as opposed to simple intersections among plates (Buntine, 1994; Gilks et al., 1994).

Just as the relational schema is a template for skeletons, the structure of a relational model can be viewed as a template for ground graphs: dependencies applied to skeletons.

Definition 4.9 (Ground graph) The *ground graph* $GG_{\mathcal{M}\sigma} = (V, E)$ for relational model structure $\mathcal{M} = (\mathcal{S}, \mathcal{D})$ and skeleton $\sigma \in \Sigma_{\mathcal{S}}$ is a directed graph with nodes $V = \{i.X \mid I \in \mathcal{E} \cup \mathcal{R} \ \wedge \ X \in \mathcal{A}(I) \ \wedge \ i \in \sigma(I)\}$ and edges $E = \{i_k.Y \rightarrow i_j.X \mid i_k.Y, i_j.X \in V \ \wedge \ i_k.Y \in [I_j, \ldots, I_k].Y|_{i_j} \ \wedge \ [I_j, \ldots, I_k].Y \rightarrow [I_j].X \in \mathcal{D}\}$.

A ground graph is a directed graph with (1) a node (random variable) for each attribute of every entity and relationship instance in a skeleton and (2) an edge from $i_k.Y$ to $i_j.X$ if they belong to the parent and child relational variable instances, respectively, of some dependency in the model. The concept of a ground graph appears for any type of relational model, graphical or logic-based. For example, PRMs produce "ground Bayesian networks" that are structurally equivalent to ground graphs, and Markov logic networks yield ground Markov networks by applying all formulas to a set of constants (Richardson and Domingos, 2006). The example ground graph shown in Figure 2.2(b) is the result of applying the dependencies in the relational model shown in Figure 2.2(a) to the skeleton in Figure 2.1(b).

Similar to Bayesian networks, given the parameters of a relational model, a *parameterized ground graph* can express a joint distribution that factors as a product of the conditional distributions:

$$P(GG_{\mathcal{M}_\Theta \sigma}) = \prod_{I \in \mathcal{E} \cup \mathcal{R}} \prod_{X \in \mathcal{A}(I)} \prod_{i \in \sigma(I)} P\big(i.X \mid \mathit{parents}(i.X)\big)$$

where each $i.X$ is assigned the conditional distribution defined for $[I].X$ (a process referred to as parameter-tying).

Relational models only define coherent joint probability distributions if they produce acyclic ground graphs. A useful construct for checking model acyclicity is the class dependency graph (Getoor et al., 2007), defined as:

Definition 4.10 (Class dependency graph) The class dependency graph $G_{\mathcal{M}} = (V, E)$ for relational model structure $\mathcal{M} = (\mathcal{S}, \mathcal{D})$ is a directed graph with a node for each attribute of every item class $V = \big\{I.X \mid I \in \mathcal{E} \cup \mathcal{R} \wedge X \in \mathcal{A}(I)\big\}$ and edges between pairs of attributes supported by relational dependencies in the model $E = \big\{I_k.Y \rightarrow I_j.X \mid [I_j, \ldots, I_k].Y \rightarrow [I_j].X \in \mathcal{D}\big\}$.

If the relational dependencies form an acyclic class dependency graph, then every possible ground graph of that model is acyclic as well (Getoor et al., 2007). Given an acyclic relational model, the ground graph has the same semantics as a Bayesian network (Getoor, 2001; Heckerman et al., 2007). All future references to acyclic relational models refer to relational models whose structure forms acyclic class dependency graphs.

By Lemma 4.1 and Definition 4.9, one relational dependency may imply dependence between the instances of many relational variables. If there is an edge from $i_k.Y$ to $i_j.X$ in the ground graph, then there is an implied dependency between all relational variables for which $i_k.Y$ and $i_j.X$ are elements of their instances.

Example 4.9 The relational dependency [Employee].*Competence*→[Employee].*Salary* yields the edge Roger.*Competence* → Roger.*Salary* in the ground graph of Figure 2.2(b) because Roger.*Competence* ∈ [Employee].*Competence*$|_{\text{Roger}}$. However, Roger.*Competence* ∈ [Employee, Develops, Product, Develops, Employee].*Competence*$|_{\text{Sally}}$ (as is Roger.*Salary*, replacing *Competence* with *Salary*). Consequently, the relational dependency *implies* dependence among the random variables in the instances of [Employee, Develops, Product, Develops, Employee].*Competence* and [Employee, Develops, Product, Develops, Employee].*Salary*. □

These implied dependencies form the crux of the challenge of identifying independence in relational models. Additionally, the intersection between the terminal sets of two relational paths is crucial for reasoning about independence because a random variable can belong to the instances of more than one relational variable. Since d-separation only guarantees independence when there are no d-connecting paths, we must consider all possible paths between pairs of random variables, either of which may be a member of multiple relational variable instances. In Section 5, we define relational d-separation and provide an appropriate representation, the abstract ground graph, that enables straightforward reasoning about d-separation.

5. Relational *d*-Separation

Conditional independence facts are correctly entailed by the rules of d-separation, but only when applied to the graphical structure of Bayesian networks. Every ground graph of a Bayesian network consists of a set of identical copies of the model structure (see Appendix D). Thus, the implications of d-separation on Bayesian networks hold for all instances in every ground graph. In contrast, the structure of a relational model is a template for ground graphs, and the structure of a ground graph varies with the underlying skeleton (which is typically more complex than a set of disconnected instances). Conditional independence facts are only useful when they hold across all ground graphs that are consistent with the model, which leads to the following definition:

Definition 5.1 (Relational *d*-separation) Let $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ be three distinct sets of relational variables with the same perspective $B \in \mathcal{E} \cup \mathcal{R}$ defined over relational schema $\mathcal{S}$. Then, for relational model structure $\mathcal{M}$, $\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ if and only if, for all skeletons $\sigma \in \Sigma_{\mathcal{S}}$, $\mathbf{X}|_b$ and $\mathbf{Y}|_b$ are d-separated by $\mathbf{Z}|_b$ in ground graph $GG_{\mathcal{M}\sigma}$ for all $b \in \sigma(B)$.

For any relational d-separation query, it is necessary that all relational variables in $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ have the same perspective (otherwise, the query would be incoherent).[5] For $\mathbf{X}$ and $\mathbf{Y}$ to be d-separated by $\mathbf{Z}$ in relational model structure $\mathcal{M}$, d-separation must hold for all instantiations of those relational variables for all possible skeletons. This is a conservative definition, but it is consistent with the semantics of d-separation on Bayesian networks—it guarantees independence, but it does not guarantee dependence. If there exists even one skeleton and faithful distribution represented by the relational model for which $\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y} \mid \mathbf{Z}$, then $\mathbf{X}$ and $\mathbf{Y}$ are not d-separated by $\mathbf{Z}$.

Given the semantics specified in Definition 5.1, answering relational d-separation queries is challenging for several reasons:

D-separation must hold over all ground graphs: The implications of d-separation on Bayesian networks hold for all possible ground graphs. However, the ground graphs of a Bayesian network consist of identical copies of the structure of the model, and it is sufficient to reason about d-separation on a single subgraph. Although it is possible to verify d-separation on a single ground graph of a relational model, the conclusion may not generalize, and ground graphs can be arbitrarily large.

Relational models are templates: The structure of a relational model is a directed acyclic graph, but the dependencies are actually templates for constructing ground graphs. The rules of d-separation do not directly apply to relational models, only to their ground graphs. Applying the rules of d-separation to a relational model frequently leads to incorrect conclusions because of unrepresented d-connecting paths that are only manifest in ground graphs.

Instances of relational variables may intersect: The instances of two different relational variables may have non-empty intersections, as described by Lemma 4.1. These intersections may be involved in relationally d-connecting paths, such as the example in Section 2. As a result, a sound and complete approach to answering relational d-separation queries must account for these paths.

Relational models may be specified from multiple perspectives: Relational models are defined by relational dependencies, each specified from a single perspective. However, variables in a ground graph may contribute to *multiple* relational variable instances, each defined from a *different* perspective. Thus, reasoning about implied dependencies between arbitrary relational variables, such as the one described in Example 4.9, requires a method to translate dependencies across perspectives.

5.1 Abstracting over All Ground Graphs

The definition of relational d-separation and its challenges suggest a solution that abstracts over all possible ground graphs and explicitly represents the potential intersection between

5. This trivially holds for d-separation in Bayesian networks as all "propositional" variables have the same implicit perspective.

pairs of relational variable instances. We introduce a new lifted representation, called the *abstract ground graph*, that captures all dependencies among arbitrary relational variables for all ground graphs, using knowledge of only the schema and the model. To represent all dependencies, the construction of an abstract ground graph uses the *extend* method, which maps a relational dependency to a set of implied dependencies for different perspectives. Each abstract ground graph of a relational model is defined with respect to a given perspective and can be used to reason about relational d-separation queries for that perspective.

Definition 5.2 (Abstract ground graph) An *abstract ground graph* $AGG_{\mathcal{M}B} = (V, E)$ for relational model structure $\mathcal{M} = (\mathcal{S}, \mathcal{D})$ and perspective $B \in \mathcal{E} \cup \mathcal{R}$ is a directed graph that abstracts the dependencies $\mathcal{D}$ for all ground graphs $GG_{\mathcal{M}\sigma}$, where $\sigma \in \Sigma_{\mathcal{S}}$.

The set of nodes in $AGG_{\mathcal{M}B}$ is $V = RV \cup IV$, where

- RV is the set of all relational variables of the form $[B, \ldots, I_j].X$
- IV is the set of all pairs of relational variables that could have non-empty intersections (referred to as intersection variables):

$$\big\{RV_1 \cap RV_2 \mid RV_1, RV_2 \in RV \;\wedge\; RV_1 = [B, \ldots, I_k, \ldots, I_j].X \\ \wedge\; RV_2 = [B, \ldots, I_l, \ldots, I_j].X \;\wedge\; I_k \neq I_l\big\}$$

The set of edges in $AGG_{\mathcal{M}B}$ is $E = RVE \;\cup\; IVE$, where

- $RVE \subset RV \times RV$ is the set of edges between pairs of relational variables:

$$RVE = \big\{[B, \ldots, I_k].Y \to [B, \ldots, I_j].X \mid [I_j, \ldots, I_k].Y \to [I_j].X \in \mathcal{D} \;\wedge \\ [B, \ldots, I_k] \in \textit{extend}([B, \ldots, I_j], [I_j, \ldots, I_k])\big\}$$

- $IVE \subset IV \times RV \;\cup\; RV \times IV$ is the set of edges inherited from both relational variables involved in every intersection variable in IV:

$$IVE = \big\{\hat{Y} \to [B, \ldots, I_j].X \mid \hat{Y} = P_1.Y \cap P_2.Y \in IV \;\wedge \\ (P_1.Y \to [B, \ldots, I_j].X \in RVE \;\vee \\ P_2.Y \to [B, \ldots, I_j].X \in RVE)\big\}$$

$$\bigcup$$

$$\big\{[B, \ldots, I_k].Y \to \hat{X} \mid \hat{X} = P_1.X \cap P_2.X \in IV \;\wedge \\ ([B, \ldots, I_k].Y \to P_1.X \in RVE \;\vee \\ [B, \ldots, I_k].Y \to P_2.X \in RVE)\big\}$$

The *extend* method is described in Definition 5.3 below. Essentially, the construction of an abstract ground graph for relational model structure $\mathcal{M}$ and perspective $B \in \mathcal{E} \cup \mathcal{R}$ follows three simple steps: (1) Add a node for all relational variables from perspective B.[6] (2) Insert edges for the direct causes of every relational variable by translating the dependencies in $\mathcal{D}$ using *extend*. (3) For each pair of potentially intersecting relational variables, create a new node that inherits the direct causes and effects from both participating relational variables in the intersection. Then, to answer queries of the form "Are $\mathbf{X}$ and $\mathbf{Y}$ d-separated by

Z?" simply (1) augment **X**, **Y**, and **Z** with their corresponding intersection variables that they subsume and (2) apply the rules of *d*-separation on the abstract ground graph for the common perspective of **X**, **Y**, and **Z**. Since abstract ground graphs are defined from a specific perspective, every relational model produces a *set of abstract ground graphs*, one for each perspective in its underlying schema.

Example 5.1 Figure 5.1 shows the abstract ground graph $AGG_{\mathcal{M},\,\textsc{Employee}}$ for the organization example from the Employee perspective with hop threshold $h = 6$.[7] As in Section 2, we derive an appropriate conditioning set **Z** in order to *d*-separate individual employee competence ($\mathbf{X} = \{$[Employee].*Competence*$\}$) from the revenue of the employee's funding business units ($\mathbf{Y} = \{$[Employee, Develops, Product, Funds, Business-Unit].*Revenue*$\}$). Applying the rules of *d*-separation to the abstract ground graph, we see that it is necessary to condition on both product success ([Employee, Develops, Product].*Success*) and the competence of other employees developing the same products ([Employee, Develops, Product, Develops, Employee].*Competence*). For $h = 6$, augmenting **X**, **Y**, and **Z** with their corresponding intersection variables does not result in any changes. For $h = 8$, the abstract ground graph includes a node for relational variable [Employee, Develops, Product, Develops, Employee, Develops, Product, Funds, Business-Unit].*Revenue* (the revenue of the business units funding the other products of collaborating employees) which, by Lemma 4.1, could have a non-empty intersection with [Employee, Develops, Product, Funds, Business-Unit].*Revenue*. Therefore, **Y** would also include the intersection with this other relational variable. However, for this query, the conditioning set **Z** for $h = 6$ happens to also *d*-separate at $h = 8$ (and any $h \in \mathbb{N}^0$). □

Using the algorithm devised by Geiger et al. (1990), relational *d*-separation queries can be answered in $O(|E|)$ time with respect to the number of edges in the abstract ground graph. In practice, the size of an abstract ground graph depends on the relational schema and model (e.g., the number of entity classes, the types of cardinalities, the number of dependencies—see the experiment in Section 7.1), as well as the hop threshold limiting the length of relational paths. For the example in Figure 5.1, the abstract ground graph has 7 nodes and 7 edges (including 1 intersection node with 2 edges); for $h = 8$, it would have 13 nodes and 21 edges (including 4 intersection nodes with 13 edges). Abstract ground graphs are invariant to the size of ground graphs, even though ground graphs can be arbitrarily large—that is, relational databases have no maximum size.

Next, we formally define the *extend* method, which is used internally for the construction of abstract ground graphs. This method translates dependencies specified in the model into dependencies in the abstract ground graph.

6. In theory, abstract ground graphs can have an infinite number of nodes as relational paths may have no bound. In practice, a *hop threshold* $h \in \mathbb{N}^0$ is enforced to limit the length of these paths. Hops are defined as the number of times the path "hops" between item classes in the schema, or one less than the length of the path.
7. The variables *Salary* and *Budget* are removed for simplicity. They are irrelevant for this *d*-separation example as they are solely effects of other variables.

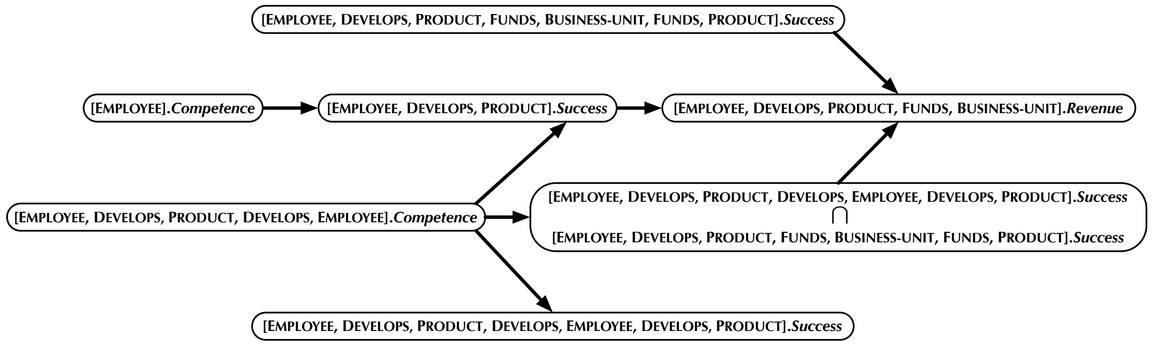

[Employee, Develops, Product, Funds, Business-Unit, Funds, Product].*Success*
[Employee].*Competence*
[Employee, Develops, Product].*Success*
[Employee, Develops, Product, Funds, Business-Unit].*Revenue*
[Employee, Develops, Product, Develops, Employee].*Competence*
[Employee, Develops, Product, Develops, Employee, Develops, Product].*Success*
∩
[Employee, Develops, Product, Funds, Business-Unit, Funds, Product].*Success*
[Employee, Develops, Product, Develops, Employee, Develops, Product].*Success*

Figure 5.1: The abstract ground graph for the organization domain model in Figure 2.2(a) from the Employee perspective with hop threshold $h = 6$ (with the variables for *Salary* and *Budget* omitted for simplicity). This abstract ground graph includes one intersection node.

Definition 5.3 (Extending relational paths) Let P_{orig} and P_{ext} be two relational paths for schema $\mathcal{S}$. The following three functions extend P_{orig} with P_{ext}:

$$extend(P_{orig}, P_{ext}) = \{P = P_{orig}^{1,n_o-i+1} + P_{ext}^{i+1,n_e} \mid i \in pivots(reverse(P_{orig}), P_{ext}) \wedge isValid(P)\}$$

$$pivots(P_1, P_2) = \{i \mid P_1^{1,i} = P_2^{1,i}\}$$

$$isValid(P) = \begin{cases} \text{True} & \text{if } P \text{ does not violate Definition 4.3} \\ \text{False} & \text{otherwise} \end{cases}$$

where n_o is the length of P_{orig}, n_e is the length of P_{ext}, $P^{i,j}$ corresponds to 1-based i-inclusive, j-inclusive subpath indexing, + is concatenation of paths, and *reverse* is a method that reverses the order of the path.

The *extend* method constructs a set of valid relational paths from two input relational paths. It first finds the indices (called pivots) of the item classes for which the input paths ($reverse(P_{orig})$ and P_{ext}) have a common starting subpath. Then, it concatenates the two input paths at each pivot, removing one of the duplicated subpaths (see Example 5.2). Since d-separation requires blocking all paths of dependence between two sets of variables, the *extend* method is critical to ensure the soundness and completeness of our approach. The abstract ground graph must capture all paths of dependence among the random variables in the relational variable instances for all represented ground graphs. However, relational model structures are specified by relational dependencies, each from a given perspective and with outcomes that have singleton relational paths. The *extend* method is called repeatedly during the creation of an abstract ground graph, with P_{orig} set to some relational path and P_{ext} drawn from the relational path of the treatment in some relational dependency.

Example 5.2 During the construction of the abstract ground graph $AGG_{\mathcal{M},\text{Employee}}$ depicted in Figure 5.1, the *extend* method is called several times. First, all relational variables from the Employee perspective are added as nodes in the graph. Next, *extend* is used to insert edges corresponding to direct causes. Consider the node for [Employee, Develops, Product].*Success*. The construction of $AGG_{\mathcal{M},\text{Employee}}$ calls $extend(P_{orig}, P_{ext})$ with P_{orig} = [Employee, Develops, Product] and P_{ext} = [Product, Develops, Employee] because [Product, Develops, Employee].*Competence* $\rightarrow$ [Product].*Success* $\in \mathcal{D}$. Here, $extend(P_{orig}, P_{ext})$ = {[Employee], [Employee, Develops, Product, Develops, Employee]}, which leads to the insertion of two edges in the abstract ground graph. Note that $pivots(reverse(P_{orig}), P_{ext}) = \{1, 2, 3\}$, and the pivot at $i = 2$ yields the invalid relational path [Employee, Develops, Employee]. □

We also describe two important properties of the *extend* method with the following two lemmas. The first lemma states that every relational path produced by *extend* yields a terminal set for some skeleton such that there is an item instance also reachable by the two original paths. This lemma is useful for proving the soundness of our abstraction: All edges inserted in an abstract ground graph correspond to edges in some ground graph.

Lemma 5.1 *Let* $P_{orig} = [I_1, \ldots, I_j]$ *and* $P_{ext} = [I_j, \ldots, I_k]$ *be two relational paths with* $\mathbf{P} = extend(P_{orig}, P_{ext})$*. Then,* $\forall P \in \mathbf{P}$ *there exists a relational skeleton* $\sigma \in \Sigma_{\mathcal{S}}$ *such that* $\exists i_1 \in \sigma(I_1)$ *such that* $\exists i_k \in P|_{i_1}$ *and* $\exists i_j \in P_{orig}|_{i_1}$ *such that* $i_k \in P_{ext}|_{i_j}$.

Proof. See Appendix A.

Example 5.3 Let σ be the skeleton shown in Figure 2.1(b), let P_{orig} = [Employee, Develops, Product], let P_{ext} = [Product, Develops, Employee], and let i_1 = Sally $\in \sigma$(Employee). From Example 5.2, we know that $\mathbf{P} = extend(P_{orig}, P_{ext})$ = {[Employee], [Employee, Develops, Product, Develops, Employee]}. We also have [Employee]$|_{\text{Sally}}$ = {Sally} and [Employee, Develops, Product, Develops, Employee]$|_{\text{Sally}}$ = {Quinn, Roger, Thomas}. By Lemma 5.1, there should exist an $i_j \in P_{orig}|_{i_1}$ such that Sally and at least one of Quinn, Roger, and Thomas are in the terminal set $P_{ext}|_{i_j}$. We have $P_{orig}|_{\text{Sally}}$ = {Laptop, Tablet}, and $P_{ext}|_{\text{Laptop}}$ = {Quinn, Roger, Sally} and $P_{ext}|_{\text{Tablet}}$ = {Sally, Thomas}. So, the lemma clearly holds for this example. □

Lemma 5.1 guarantees that, for some relational skeleton, there exists an item instance in the terminal sets produced by *extend* that also appears in the terminal set of P_{ext} via some instance in the terminal set of P_{orig}. It is also possible (although infrequent) that there exist items reachable by P_{orig} and P_{ext} that are not in the terminal set of any path produced with $extend(P_{orig}, P_{ext})$. The following lemma describes this unreachable set of items, stating that there must exist an alternative relational path P'_{orig} that intersects with P_{orig} and, when using *extend*, catches those remaining items. This lemma is important for proving the completeness of our abstraction: All edges in all ground graphs are represented in the abstract ground graph.

Lemma 5.2 *Let* $\sigma \in \Sigma_{\mathcal{S}}$ *be a relational skeleton, and let* $P_{orig} = [I_1, \ldots, I_j]$ *and* $P_{ext} = [I_j, \ldots, I_k]$ *be two relational paths with* $\mathbf{P} = extend(P_{orig}, P_{ext})$*. Then,* $\forall i_1 \in \sigma(I_1)\ \forall i_j \in P_{orig}|_{i_1}\ \forall i_k \in P_{ext}|_{i_j}$ *if* $\forall P \in \mathbf{P}\ i_k \notin P|_{i_1}$*, then* $\exists P'_{orig}$ *such that* $P_{orig}|_{i_1} \cap P'_{orig}|_{i_1} \neq \emptyset$ *and* $i_k \in P'|_{i_1}$ *for some* $P' \in extend(P'_{orig}, P_{ext})$.

Proof. See Appendix A.

Example 5.4 Although Lemma 5.2 does not apply to the organization domain as currently represented, it could apply if either (1) there were cycles in the relational schema or (2) the path specifications on the relational dependencies included a cycle. Consider additional relationships between employees and products. If employees could be involved with products at various stages (e.g., research, development, testing, marketing), then there would be alternative relational paths for which the lemma might apply. The proof of the lemma in Appendix A provides abstract conditions describing when the lemma applies. □

5.2 Proof of Correctness

The correctness of our approach to relational d-separation relies on several facts: (1) d-separation is valid for directed acyclic graphs; (2) ground graphs are directed acyclic graphs; and (3) abstract ground graphs are directed acyclic graphs that represent exactly the edges that could appear in all possible ground graphs. It follows that d-separation on abstract ground graphs, augmented by intersection variables, is sound and complete for all ground graphs.[8] Additionally, we show that since relational d-separation is sound and complete, it is also equivalent to the Markov condition for relational models. Using the previous definitions and lemmas, the following sequence of results proves the correctness of our approach to identifying independence in relational models.

Theorem 5.1 *The rules of d-separation are sound and complete for directed acyclic graphs.*

Proof. Due to Verma and Pearl (1988) for soundness and Geiger and Pearl (1988) for completeness. ■

Theorem 5.1 implies that (1) all conditional independence facts derived by d-separation on a Bayesian network structure hold in any distribution represented by that model (soundness) and (2) all conditional independence facts that hold in a faithful distribution can be inferred from d-separation applied to the Bayesian network that encodes the distribution (completeness).

Lemma 5.3 *For every acyclic relational model structure $\mathcal{M}$ and skeleton $\sigma \in \Sigma_{\mathcal{S}}$, the ground graph $GG_{\mathcal{M}\sigma}$ is a directed acyclic graph.*

Proof. Due to both Heckerman et al. (2007) for DAPER models and Getoor (2001) for PRMs. ■

By Theorem 5.1 and Lemma 5.3, d-separation is sound and complete when applied to a ground graph. However, Definition 5.1 states that relational d-separation must hold across *all possible* ground graphs, which is the reason for constructing the abstract ground graph representation.

8. In Appendix E, we provide proofs of soundness and completeness for abstract ground graphs and relational d-separation that are limited by practical hop threshold bounds.

Theorem 5.2 *For every acyclic relational model structure $\mathcal{M}$ and perspective $B \in \mathcal{E} \cup \mathcal{R}$, the abstract ground graph $AGG_{\mathcal{M}B}$ is sound and complete for all ground graphs $GG_{\mathcal{M}\sigma}$ with skeleton $\sigma \in \Sigma_{\mathcal{S}}$.*

Proof. See Appendix A.

Theorem 5.2 guarantees that, for a given perspective, an abstract ground graph captures all possible paths of dependence between any pair of variables in any ground graph. The details of the proof provide the reasons why explicitly representing intersection variables is necessary for ensuring a sound and complete abstraction.

Theorem 5.3 *For every acyclic relational model structure $\mathcal{M}$ and perspective $B \in \mathcal{E} \cup \mathcal{R}$, the abstract ground graph $AGG_{\mathcal{M}B}$ is directed and acyclic.*

Proof. See Appendix A.

Theorem 5.3 ensures that the standard rules of d-separation can apply directly to abstract ground graphs because they are acyclic given an acyclic model. We now have sufficient supporting theory to prove that d-separation on abstract ground graphs is sound and complete. In the following theorem, we define $\bar{\mathbf{W}}$ as the set of nodes augmented with their corresponding intersection nodes for the set of relational variables $\mathbf{W}$: $\bar{\mathbf{W}} = \mathbf{W} \cup \bigcup_{W \in \mathbf{W}} \{W \cap W' \mid W \cap W' \text{ is an intersection node in } AGG_{\mathcal{M}B}\}$.

Theorem 5.4 *Relational d-separation is sound and complete for abstract ground graphs. Let $\mathcal{M}$ be an acyclic relational model structure, and let $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ be three distinct sets of relational variables for perspective $B \in \mathcal{E} \cup \mathcal{R}$ defined over relational schema $\mathcal{S}$. Then, $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ are d-separated by $\bar{\mathbf{Z}}$ on the abstract ground graph $AGG_{\mathcal{M}B}$ if and only if for all skeletons $\sigma \in \Sigma_{\mathcal{S}}$ and for all $b \in \sigma(B)$, $\mathbf{X}|_b$ and $\mathbf{Y}|_b$ are d-separated by $\mathbf{Z}|_b$ in ground graph $GG_{\mathcal{M}\sigma}$.*

Proof. We must show that d-separation on an abstract ground graph implies d-separation on all ground graphs it represents (soundness) and that d-separation facts that hold across all ground graphs are also entailed by d-separation on the abstract ground graph (completeness).

Soundness: Assume that $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ are d-separated by $\bar{\mathbf{Z}}$ on $AGG_{\mathcal{M}B}$. Assume for contradiction that there exists an item instance $b \in \sigma(B)$ such that $\mathbf{X}|_b$ and $\mathbf{Y}|_b$ are *not* d-separated by $\mathbf{Z}|_b$ in the ground graph $GG_{\mathcal{M}\sigma}$ for some arbitrary skeleton σ. Then, there must exist a d-connecting path p from some $x \in \mathbf{X}|_b$ to some $y \in \mathbf{Y}|_b$ given all $z \in \mathbf{Z}|_b$. By Theorem 5.2, $AGG_{\mathcal{M}B}$ is complete, so all edges in $GG_{\mathcal{M}\sigma}$ are captured by edges in $AGG_{\mathcal{M}B}$. So, path p must be represented from some node in $\{N_x \mid x \in N_x|_b\}$ to some node in $\{N_y \mid y \in N_y|_b\}$, where N_x, N_y are nodes in $AGG_{\mathcal{M}B}$. If p is d-connecting in $GG_{\mathcal{M}\sigma}$, then it is d-connecting in $AGG_{\mathcal{M}B}$, implying that $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ are *not* d-separated by $\bar{\mathbf{Z}}$. So, $\mathbf{X}|_b$ and $\mathbf{Y}|_b$ must be d-separated by $\mathbf{Z}|_b$.

Completeness: Assume that $\mathbf{X}|_b$ and $\mathbf{Y}|_b$ are d-separated by $\mathbf{Z}|_b$ in the ground graph $GG_{\mathcal{M}\sigma}$ for all skeletons σ for all $b \in \sigma(B)$. Assume for contradiction that $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ are *not* d-separated by $\bar{\mathbf{Z}}$ on $AGG_{\mathcal{M}B}$. Then, there must exist a d-connecting path p for some relational variable $X \in \bar{\mathbf{X}}$ to some $Y \in \bar{\mathbf{Y}}$ given all $Z \in \bar{\mathbf{Z}}$. By Theorem 5.2, $AGG_{\mathcal{M}B}$ is sound, so every edge in $AGG_{\mathcal{M}B}$ must correspond to some pair of variables in some ground graph. So, if p is d-connecting in $AGG_{\mathcal{M}B}$, then there must exist some skeleton σ such that p is d-connecting in $GG_{\mathcal{M}\sigma}$ for some $b \in \sigma(B)$, implying that d-separation does not hold for that ground graph. So, $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ must be d-separated by $\bar{\mathbf{Z}}$ on $AGG_{\mathcal{M}B}$. ■

Theorem 5.4 proves that d-separation on abstract ground graphs is a sound and complete solution to identifying independence in relational models. Theorem 5.1 also implies that the set of conditional independence facts derived from abstract ground graphs is exactly the same as the set of conditional independencies that all distributions represented by all possible ground graphs have in common.

Corollary 5.1 $\bar{\mathbf{X}}$ *and* $\bar{\mathbf{Y}}$ *are d-connected given* $\bar{\mathbf{Z}}$ *on the abstract ground graph* $AGG_{\mathcal{M}B}$ *if and only if there exists a skeleton* $\sigma \in \Sigma_{\mathcal{S}}$ *and an item instance* $b \in \sigma(B)$ *such that* $\mathbf{X}|_b$ *and* $\mathbf{Y}|_b$ *are d-connected given* $\mathbf{Z}|_b$ *in ground graph* $GG_{\mathcal{M}\sigma}$.

Corollary 5.1 is logically equivalent to Theorem 5.4. While a simple restatement of the previous theorem, it is important to emphasize that relational d-separation claims d-connection if and only if there exists a ground graph for which $X|_b$ and $Y|_b$ are d-connected given $\mathbf{Z}|_b$. This implies that there may be some ground graphs for which $X|_b$ and $Y|_b$ are d-separated by $\mathbf{Z}|_b$, but the abstract ground graph still claims d-connection. This may happen if the relational skeleton does not enable certain underlying relational connections. For example, if the relational skeleton in Figure 2.1(b) included only products that were developed by a single employee, then there would be no relationally d-connecting path in the example in Section 2. If this is a fundamental property of the domain (e.g., there are products developed by a single employee and products developed by multiple employees), then revising the underlying schema to include two different classes of products would yield a more accurate model implying a larger set of conditional independencies.

Additionally, we can show that relational d-separation is equivalent to the Markov condition on relational models.

Definition 5.4 (Relational Markov condition) Let X be a relational variable for perspective $B \in \mathcal{E} \cup \mathcal{R}$ defined over relational schema $\mathcal{S}$. Let $nd(X)$ be the non-descendant variables of X, and let $pa(X)$ be the set of parent variables of X. Then, for relational model $\mathcal{M}_\Theta$, $P\big(X \mid nd(X), pa(X)\big) = P\big(X \mid pa(X)\big)$ if and only if $\forall x \in X|_b \; P\big(x \mid nd(x), pa(x)\big) = P\big(x \mid pa(x)\big)$ in parameterized ground graph $GG_{\mathcal{M}_\Theta \sigma}$ for all skeletons $\sigma \in \Sigma_{\mathcal{S}}$ and for all $b \in \sigma(B)$.

In other words, a relational variable X is independent of its non-descendants given its parents if and only if, for all possible parameterized ground graphs, the Markov condition holds for all instances of X. For Bayesian networks, the Markov condition is equivalent to d-separation (Neapolitan, 2004). Because parameterized ground graphs are Bayesian networks (implied by Lemma 5.3) and relational d-separation on abstract ground graphs is sound and complete (by Theorem 5.4), we can conclude that relational d-separation is equivalent to the relational Markov condition.

6. Naïve Relational *d*-Separation Is Frequently Incorrect

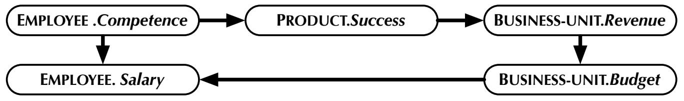
If the rules of d-separation for Bayesian networks were applied directly to the structure of relational models, how frequently would the conditional independence conclusions be correct? In this section, we evaluate the necessity of our approach—relational d-separation executed on abstract ground graphs. We empirically compare the consistency of a naïve approach against our sound and complete solution over a large space of synthetic causal models. To promote a fair comparison, we restrict the space of relational models to those with underlying dependencies that could feasibly be represented and recovered by a naïve approach. We describe this space of models, present a reasonable approach for applying traditional d-separation to the structure of relational models, and quantify its decrease in expressive power and accuracy.

Consider the following limited definition of relational paths, which itself limits the space of models and conditional independence queries. A *simple relational path* $P = [I_j, \ldots, I_k]$ for relational schema $\mathcal{S}$ is a relational path such that $I_j \neq \cdots \neq I_k$. The sole difference between relational paths (Definition 4.3) and simple relational paths is that no item class may appear more than once along the latter. This yields paths drawn directly from a schema diagram. For the example in Figure 2.1(a), [Employee, Develops, Product] is simple whereas [Employee, Develops, Product, Develops, Employee] is not.

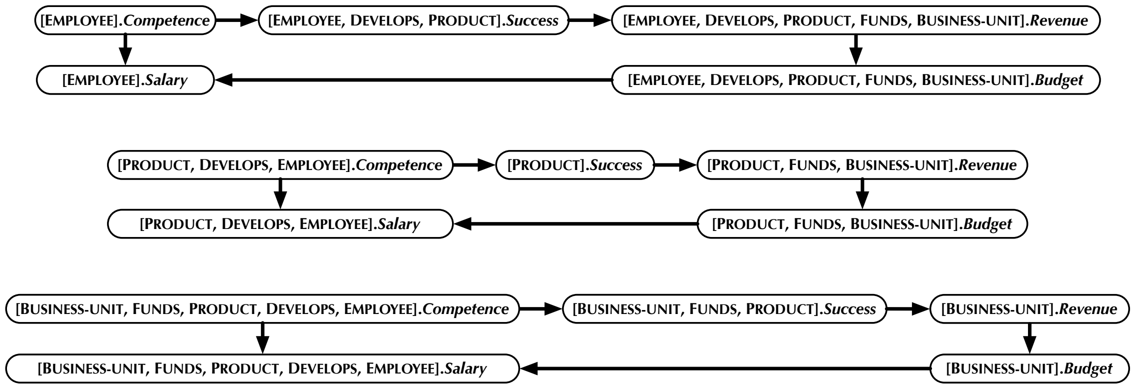
Additionally, we define *simple relational schemas* such that, for any two item classes $I_j, I_k \in \mathcal{E} \cup \mathcal{R}$, there exists at most one simple relational path between them (i.e., no cycles occur in the schema diagram). The example in Figure 2.1(a) is a simple relational schema. The restriction to simple relational paths and schemas yields similar definitions for *simple relational variables*, *simple relational dependencies*, and *simple relational models*. The relational model in Figure 2.2(a) is simple because it includes only simple relational dependencies.

A first approximation to relational d-separation would be to apply the rules of traditional d-separation directly to the graphical representation of relational models. This is equivalent to applying d-separation to the class dependency graph $G_{\mathcal{M}}$ (see Definition 4.10) of relational model $\mathcal{M}$. The class dependency graph for the model in Figure 2.2(a) is shown in Figure 6.1(a). Note that the class dependency graph ignores path designators on dependencies, does not include all implications of dependencies among arbitrary relational variables, and does not represent intersection variables.

Although the class dependency graph is independent of perspectives, testing any conditional independence fact *requires* choosing a perspective. All relational variables must have a common base item class; otherwise, no method can produce a single consistent, propositional table from a relational database. For example, consider the construction of a table describing employees with columns for their salary, the success of products they develop, and the revenue of the business units they operate under. This procedure requires joining the instances of three relational variables ([Employee].*Salary*, [Employee, Develops, Product].*Success*, and [Employee, Develops, Product, Funds, Business-Unit].*Revenue*) for every common base item instance, from Paul to Thomas. See, for example, the resulting propositional table for these relational variables and an example query in Table D.1 and Figure D.2, respectively. An individual relational variable requires joining the item classes within its relational path, but combining a collection of relational variables requires joining



(a)


[EMPLOYEE].Competence
[EMPLOYEE, DEVELOPS, PRODUCT].Success
[EMPLOYEE, DEVELOPS, PRODUCT, FUNDS, BUSINESS-UNIT].Revenue
[EMPLOYEE].Salary
[EMPLOYEE, DEVELOPS, PRODUCT, FUNDS, BUSINESS-UNIT].Budget
[PRODUCT, DEVELOPS, EMPLOYEE].Competence
[PRODUCT].Success
[PRODUCT, FUNDS, BUSINESS-UNIT].Revenue
[PRODUCT, DEVELOPS, EMPLOYEE].Salary
[PRODUCT, FUNDS, BUSINESS-UNIT].Budget
[BUSINESS-UNIT, FUNDS, PRODUCT, DEVELOPS, EMPLOYEE].Competence
[BUSINESS-UNIT, FUNDS, PRODUCT].Success
[BUSINESS-UNIT].Revenue
[BUSINESS-UNIT, FUNDS, PRODUCT, DEVELOPS, EMPLOYEE].Salary
[BUSINESS-UNIT].Budget

(b)

Figure 6.1: For the model in Figure 2.2(a), (a) the class dependency graph and (b) three simple abstract ground graphs for the Employee, Product, and Business-Unit perspectives.

on their common base item class. Fortunately, given a perspective and the space of simple relational schemas and models, a class dependency graph is equivalent to a *simple abstract ground graph.*

Definition 6.1 (Simple abstract ground graph) For simple relational model $\mathcal{M} = (\mathcal{S}, \mathcal{D})$ and perspective $B \in \mathcal{E} \cup \mathcal{R}$, the *simple abstract ground graph* $AGG^s_{\mathcal{M}B}$ is the directed acyclic graph (V, E) that abstracts the dependencies $\mathcal{D}$ among simple relational variables. The nodes consist of simple relational variables $\{[B, \dots, I_j].X \mid B \neq \cdots \neq I_j\}$, and the edges connect those nodes $\{[B, \dots, I_k].Y \to [B, \dots, I_j].X \mid [I_j, \dots, I_k].Y \to [I_j].X \in \mathcal{D} \ \wedge$ $[B, \dots, I_k] \in extend([B, \dots, I_j], [I_j, \dots, I_k]) \ \wedge \ [B, \dots, I_k].Y, [B, \dots, I_j].X \in V\}$.

Simple abstract ground graphs only include nodes for simple relational variables and necessarily exclude intersection variables. Lemma 4.1—which characterizes the intersection between a pair of relational paths—only applies to pairs of *simple* relational paths if the schema contains cycles, which is not the case for simple relational schemas by definition. As a result, the simple abstract ground graph for a given schema and model contains the same number of nodes and edges, regardless of perspective; the nodes simply have path designators redefined from the given perspective. Figure 6.1(b) shows three simple abstract ground graphs from distinct perspectives for the model in Figure 2.2(a). As noted above, simple abstract ground graphs are qualitatively the same as the class dependency graph, but they enable answering relational d-separation queries, which requires a common perspective in order to be semantically meaningful.

The naïve approach to relational d-separation derives conditional independence facts from simple abstract ground graphs (Definition 6.1). The sound and complete approach described in this paper applies d-separation—with input variable sets augmented by their intersection variables—to "regular" abstract ground graphs, as described by Definition 5.2. Clearly, if d-separation on a simple abstract ground graph claims that $\mathbf{X}$ is d-separated from $\mathbf{Y}$ given $\mathbf{Z}$, then d-separation on the regular abstract ground graph yields the same conclusion if and only if there are no d-connecting paths in the regular abstract ground graph. Either $\mathbf{X}$ and $\mathbf{Y}$ can be d-separated by a set of simple relational variables $\mathbf{Z}$, or they require non-simple relational variables—those involving relational paths with repeated item classes.[9]

To evaluate the necessity of regular abstract ground graphs (i.e., the additional paths involving non-simple relational variables and intersection variables), we compared the frequency of equivalence between the conclusions of d-separation on simple and regular abstract ground graphs. The two approaches are only equivalent if a minimal separating set consists entirely of simple relational variables.[10]

Thus, for an arbitrary pair of relational variables X and Y with a common perspective, we test the following on regular abstract ground graphs:

1. Is either X or Y a non-simple relational variable?
2. Are X and Y marginally independent?
3. Does a minimal separating set $\mathbf{Z}$ d-separate X and Y, where $\mathbf{Z}$ consists solely of simple relational variables?
4. Is there any separating set $\mathbf{Z}$ that d-separates X and Y?

If the answer to (1) is yes, then the naïve approach cannot apply since either X or Y is undefined for the simple abstract ground graph. If the answer to (2) is yes, then there is equivalence; this is a trivial case because there are no d-connecting paths for $\mathbf{Z} = \emptyset$. If the answer to (3) is yes, then there is a minimal separating set $\mathbf{Z}$ consisting of only simple relational variables. In this case, the simple abstract ground graph is sufficient, and we also have equivalence. If the answer to (4) is no, then no separating set $\mathbf{Z}$, simple or otherwise, renders X and Y conditionally independent.

We randomly generated simple relational schemas and models for 100 trials for each setting using the following parameters:

- Number of entity classes, ranging from 1 to 4.
- Number of relationship classes, fixed at one less than the number of entities, ensuring simple, connected relational schemas. Relationship cardinalities are chosen uniformly at random.
- Number of attributes for each entity and relationship class, randomly drawn from a shifted Poisson distribution with $\lambda = 1.0$ ($\sim Pois(1.0) + 1$).
- Number of dependencies in the model, ranging from 1 to 10.

9. The theoretical conditions under which equivalence occurs are sufficiently complex that they provide little utility as they essentially require reconstructing the regular abstract ground graph and checking a potentially exponential number of dependency paths.

10. If $\mathbf{X}$ and $\mathbf{Y}$ are d-separated given $\mathbf{Z}$, then $\mathbf{Z}$ is a *separating set* for $\mathbf{X}$ and $\mathbf{Y}$. A separating set $\mathbf{Z}$ is *minimal* if there is no proper subset of $\mathbf{Z}$ that is also a separating set.

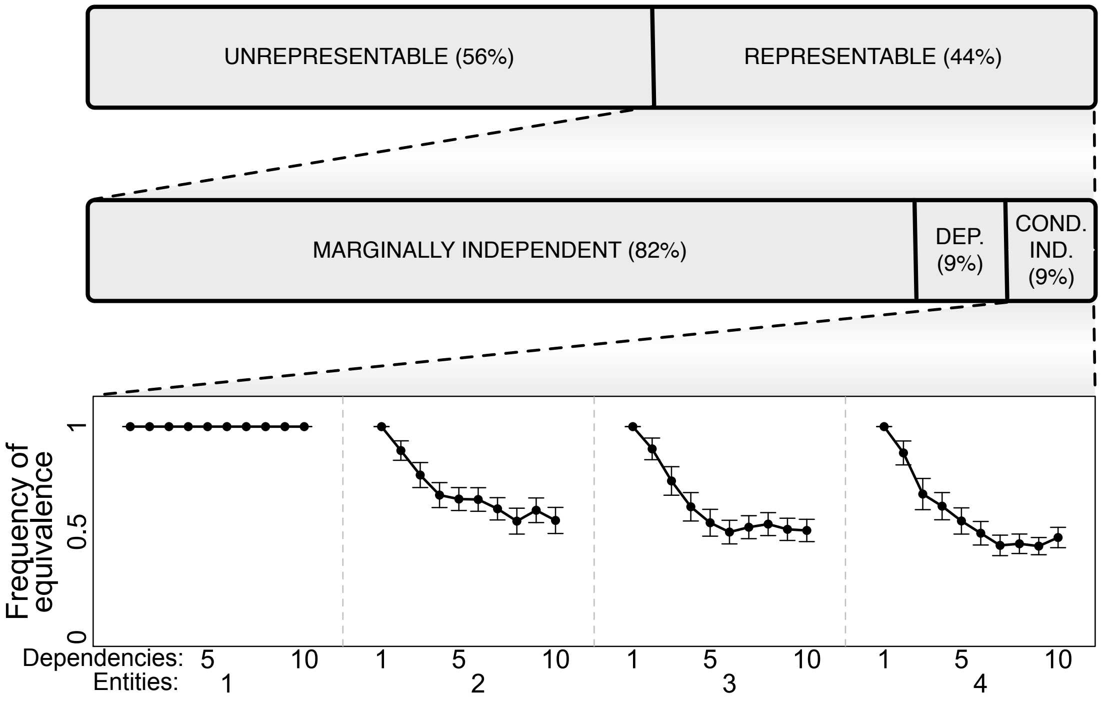


Figure 6.2: The majority (56%) of generated relational d-separation queries are not representable with the naïve approach. Of the 44% that are representable (involving only simple relational variables), 82% are marginally independent and 9% are dependent. Pairs of relational variables in the remaining 9% are conditionally independent given a non-empty separating set ($X \perp\!\!\!\perp Y \mid \mathbf{Z}$, where $\mathbf{Z} \neq \emptyset$). We test whether the *conditioning set* consists solely of simple relational variables. If so, then the naïve approach to relational d-separation is equivalent to d-separation on fully specified abstract ground graphs. This graph plots the frequency of equivalence across schemas with increasing numbers of entity classes (1–4) for varying numbers of dependencies (1–10). For schemas with more than one entity class, the frequency of equivalence decreases as the number of dependencies increases. Shown with 95% confidence intervals.

Then, for all pairs of relational variables with a common perspective limited by a hop threshold of $h = 4$, we ran the aforementioned tests against the regular abstract ground graph, limiting its relational variables by a hop threshold of $h = 8$ (a sufficient hop threshold for soundness and completeness—see Appendix E).

This procedure generated a total of almost 3.6 million pairs of relational variables to test. Approximately 56% included a non-simple relational variable; the naïve approach cannot be used to derive a conditional independence statement in these cases, requiring the full abstract ground graph in order to represent these variables. Of the remaining 44% (roughly 1.6 million), 82% were marginally independent, and 9% were not conditionally independent given any conditioning set $\mathbf{Z}$. Then, of the remaining 9% (roughly 145,000), we can test the frequency of equivalence for conditional independence facts with non-empty separating sets—the proportion of cases for which only simple relational variables are required in a minimal separating set **Z**.

Figure 6.2 shows this frequency for schemas of increasing numbers of entity classes (1–4) for varying numbers of dependencies in the causal model (1–10). Since relational schemas with a single entity class and no relationships describe propositional data, the simple abstract ground graph is equivalent to the full abstract ground graph, which is also equivalent to the model itself. In this case, the naïve approach is always equivalent because it is exactly d-separation on Bayesian networks. For truly relational schemas (with more than one entity class and at least one relationship class), the frequency of equivalence decreases as the number of dependencies in the model increases. Additionally, the frequency of equivalence decreases more as the number of entities in the schema increases. For example, the frequency of equivalence for nine dependencies is 60.3% for two entities, 51.2% for three entities, and 43.2% for four entities.

We also learned statistical models that predict the number of equivalent and non-equivalent statements in order to identify key factors that affect the frequency of equivalence. We found that the number of dependencies and size of the relational model (regulated by the number of entities and MANY cardinalities) dictate the equivalence. As a relational model deviates from a Bayesian network, we should expect more d-connecting paths in the regular but not simple abstract ground graph. This property also depends on the specific combination of dependencies in the model. Appendix F presents details of this analysis.

This experiment suggests that applying traditional d-separation directly to a relational model structure will frequently derive incorrect conditional independence facts. Additionally, there is a large class of conditional independence queries involving non-simple variables for which such an approach is undefined. These results indicate that fully specifying abstract ground graphs and applying d-separation augmented with intersection variables (as described in Section 5) is critical for accurately deriving most conditional independence facts from relational models.

7. Experiments

To complement the theoretical results, we present three experiments on synthetic data. The primary goal of these empirical results is to demonstrate the feasibility of applying relational d-separation in practice. The experiment in Section 7.1 describes the factors that influence the size of abstract ground graphs and thus the computational complexity of relational d-separation. The experiment in Section 7.2 evaluates the growth rate of separating sets produced by relational d-separation as abstract ground graphs become large. The results indicate that minimal separating sets grow much more slowly than abstract ground graphs. The experiment in Section 7.3 tests how the expectations of the relational d-separation theory match statistical conclusions on simulated data. As expected from the proofs of correctness in Section 5.2, the results indicate a close match, aside from Type I errors and certain biases of conventional statistical tests on relational data.

7.1 Abstract Ground Graph Size

Relational d-separation is executed on abstract ground graphs. Consequently, it is important to quantify the size of abstract ground graphs and identify which factors influence their

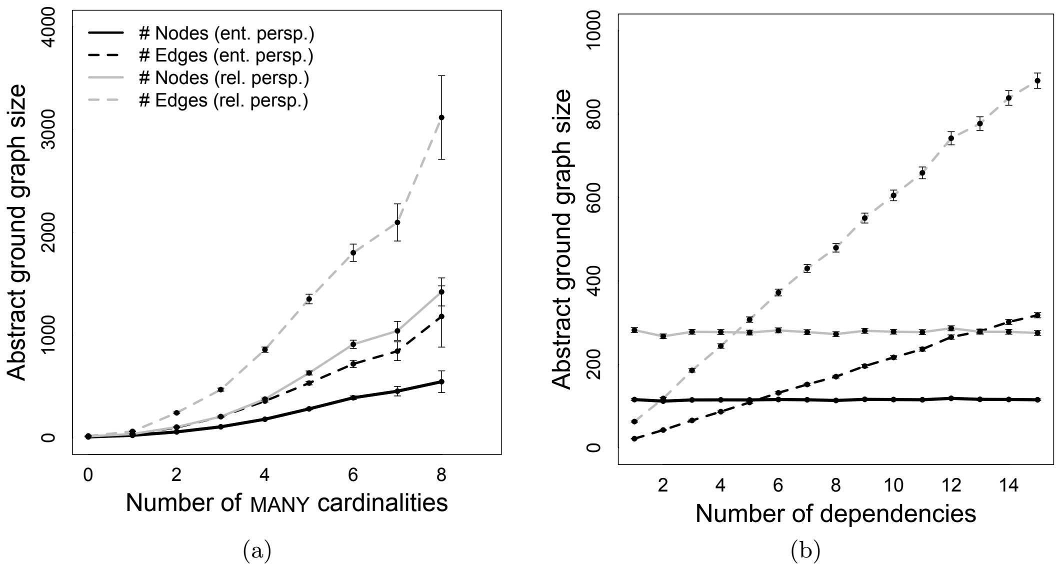


Figure 7.1: Variation of abstract ground graph size as (a) the number of MANY cardinalities in the schema increases (dependencies fixed at 10) and (b) the number of dependencies increases. Shown with 95% confidence intervals.

size. We randomly generated relational schemas and models for 1,000 trials of each setting using the following parameters:

- Number of entity classes, ranging from 1 to 4.
- Number of relationship classes, ranging from 0 to 4. The schema is guaranteed to be fully connected and includes at most a single relationship between a pair of entities. Relationship cardinalities are selected uniformly at random.
- Number of attributes for each entity and relationship class, randomly drawn from a shifted Poisson distribution with $\lambda = 1.0$ ($\sim Pois(1.0) + 1$).
- Number of dependencies in the model, ranging from 1 to 15.

This procedure generated a total of 450,000 abstract ground graphs, which included every perspective (all entity and relationship classes) for each experimental combination. We measure size as the number of nodes and edges in a given abstract ground graph. Figure 7.1(a) depicts how the size of abstract ground graphs varies with respect to the number of MANY cardinalities in the schema (fixed for models with 10 dependencies), and Figure 7.1(b) shows how it varies with respect to the number of dependencies in the model. Recall that for a single entity, abstract ground graphs are equivalent to Bayesian networks.

To determine the most influential factors of abstract ground graph size, we ran log-linear regression using independent variables that describe only the schema and model. Detailed results are provided in Appendix G. This analysis indicates that (1) as the number of entities, relationships, attributes, and MANY cardinalities increases, the number of nodes and edges grows at an exponential rate. (2) As the number of dependencies in the model increases,

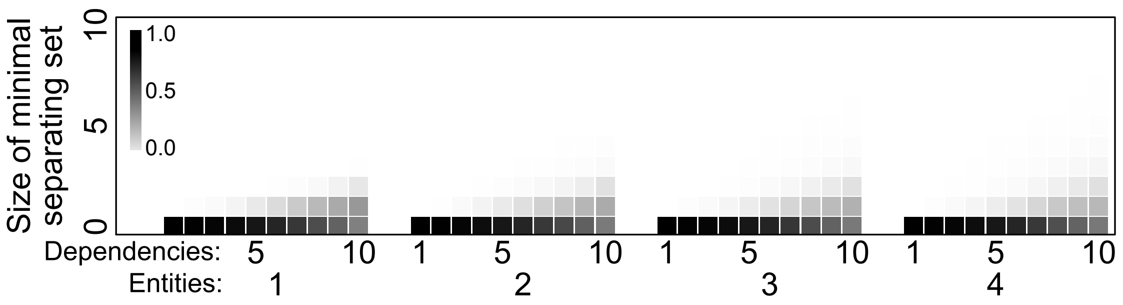


Figure 7.2: Minimal separating sets have reasonable sizes, growing only with the size of the schema and the model density. In this experiment, 99.9% of variable pairs have a minimal separating set with five or fewer variables.

the number of edges increases linearly, but the number of nodes remains invariant. And (3) abstract ground graphs for relationship perspectives are larger than entity perspectives because more relational variables can be defined.

7.2 Minimal Separating Set Size

Because abstract ground graphs can become large, one might expect that separating sets could also grow to impractical sizes. Fortunately, relational d-separation produces minimal separating sets that are empirically observed to be small. We ran 1,000 trials of each setting using the following parameters:

- Number of entity classes, ranging from 1 to 4.
- Number of relationship classes, fixed at one less than the number of entities. Relationship cardinalities are selected uniformly at random.
- Total number of attributes across entity and relationship classes, fixed at 10.
- Number of dependencies in the model, ranging from 1 to 10.

For each relational model, we identified a single minimal separating set for up to 100 randomly chosen pairs of conditionally independent relational variables. This procedure generated almost 2.5 million pairs of variables.

To identify a minimal separating set between relational variables X and Y, we modified Algorithm 4 devised by Tian et al. (1998) by starting with all parents of $\bar{X}$ and $\bar{Y}$, the variables augmented with the intersection variables they subsume in the abstract ground graph. While the discovered separating sets are *minimal*, they are not necessarily of *minimum* size because of the greedy process for removing conditioning variables from the separating set. Figure 7.2 shows the frequency of separating set size as both the number of entities and dependencies vary. In summation, roughly 83% of the pairs are marginally independent (having empty separating sets), 13% have separating sets of size one, and less than 0.1% have separating sets with more than five variables. The experimental results indicate that separating set size is strongly influenced by model density, primarily because the number of potential d-connecting paths increases as the number of dependencies increases.

7.3 Empirical Validity

As a practical demonstration, we examined how the expectations of the relational d-separation theory match the results of statistical tests on actual data. We use a standard procedure for empirically measuring internal validity of algorithms. In this case, we (1) randomly generate a relational schema, (2) randomly generate a relational model structure for that schema, (3) parameterize the model structure, (4) generate synthetic data according to the model structure and parameters, (5) randomly choose relational d-separation queries according to the known ground-truth model, and (6) compare the model theory (i.e., the d-separation conclusions) against corresponding statistical tests of conditional independence.

For steps (1) and (2), we randomly generated a relational schema $\mathcal{S}$ and relational model structure $\mathcal{M}$ for $\mathcal{S}$ for 100 trials using the following settings:

- Number of entity classes, ranging from 1 to 4.
- Number of relationship classes, fixed at one less than the number of entities. Relationship cardinalities are selected uniformly at random.
- Number of attributes for each entity and relationship class, randomly drawn from a shifted Poisson distribution with $\lambda = 1.0$ ($\sim Pois(1.0) + 1$).
- Number of dependencies in the model, fixed at 10.

Dependencies were selected greedily, choosing each one uniformly at random, subject to a maximum of 3 parent relational variables for each attribute $[I_j].X$ and enforcing acyclicity of the model structure.

For step (3), we parameterized relational models using simple additive linear equations with independent, normally distributed error and the average aggregate for relational variable instances. For each attribute $[I_j].X$, we assign a conditional probability distribution

$$\sum_{[I_j,\dots,I_k].Y \in parents([I_j].X)} \big(\beta \cdot avg([I_j, \dots, I_k].Y)\big) + 0.1\epsilon$$

if $[I_j].X$ has parents, where

$$\beta = \frac{0.9}{|parents([I_j].X)|}$$

to provide equal contribution for each direct cause and $\epsilon \sim N(0, 1)$ (error drawn from a standard normal distribution). If $[I_j].X$ has no parents, its value is just drawn from ϵ.

For step (4), we first generated a relational skeleton σ (because the current model space assumes that attributes do not cause entity or relationship existence) and then populated each attribute value by drawing from its corresponding conditional distribution. Each entity class is initialized to 1,000 instances. Relationship instances were constructed via a latent homophily process, similar to the method used by Shalizi and Thomas (2011). Each entity instance received a single latent variable, marginally independent from all other variables. The probability of any relationship instance was drawn from

$$\frac{e^{-\alpha d}}{1 + e^{-\alpha d}},$$

the inverse logistic function, where $d = |L_{E_1} - L_{E_2}|$, the difference between the latent variables on the two entities, and $\alpha = 10$, set as the decay parameter. We also scaled

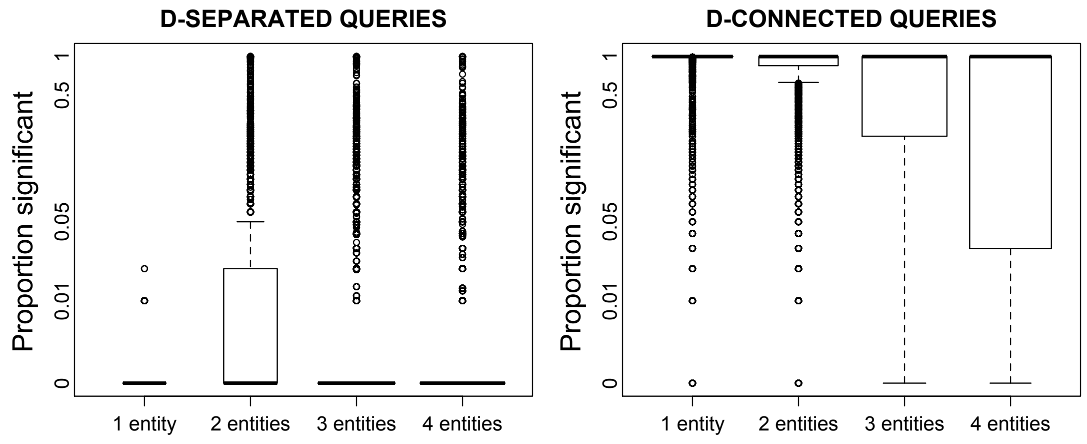


Figure 7.3: The proportion of significant trials for statistical tests of conditional independence on actual data. (Left) Evaluating queries that the model claims to be d-separated produces low rates of significant effects. (Right) Queries that the model claims are d-connected produce high rates of significant effects. Note that the generative process yields denser models for 2 entity classes since the number of dependencies is fixed at 10.

the probabilities in order to produce an expected degree of five for each entity instance when the cardinality of the relationship is MANY. Since the latent variables are marginally independent of all others, they are safely omitted from abstract ground graphs; their sole purpose is to generate relational skeletons that provide a greater probability of non-empty intersection variables as opposed to a random underlying link structure. We generated 100 independent relational skeletons and attribute values (i.e., 100 instantiated relational databases) for each schema and model.

Step (5) randomly chooses up to 100 true and false relational d-separation queries for a given model.[11] Since we have the ground-truth model, we can evaluate with our approach (abstract ground graphs and relational d-separation) whether these queries are true (d-separated) or false (d-connected). Each query is of the form $X \perp\!\!\!\perp Y \mid \mathbf{Z}$ such that X and Y are single relational variables, $\mathbf{Z}$ is a set of relational variables, Y has a singleton relational path (e.g., $[I_k].Y$), and all variables are from a common perspective. These queries correspond to testing potential direct causal dependencies in the relational model, similar to the tests used by constraint-based methods for learning relational models, such as RPC (Maier et al., 2010) and RCD (Maier et al., 2013).

Finally, step (6) tests for conditional independence for all such $\langle X, Y, \mathbf{Z} \rangle$ d-separation queries using linear regression (because the models were parameterized linearly) for each of the 100 data instantiations. Specifically, we tested the t-statistic for the coefficient of $avg(X)$ in the equation $Y = \beta_0 + \beta_1 \cdot avg(X) + \sum_{Z_i \in \mathbf{Z}} \beta_i \cdot avg(Z_i)$. For each query, we recorded two measurements:

- The average strength of effect, measured as squared partial correlation—the proportion of remaining variance of Y explained by X after conditioning on $\mathbf{Z}$

11. Depending on the properties of the schema and model, it may not always be feasible to identify 100 true or false d-separation statements.

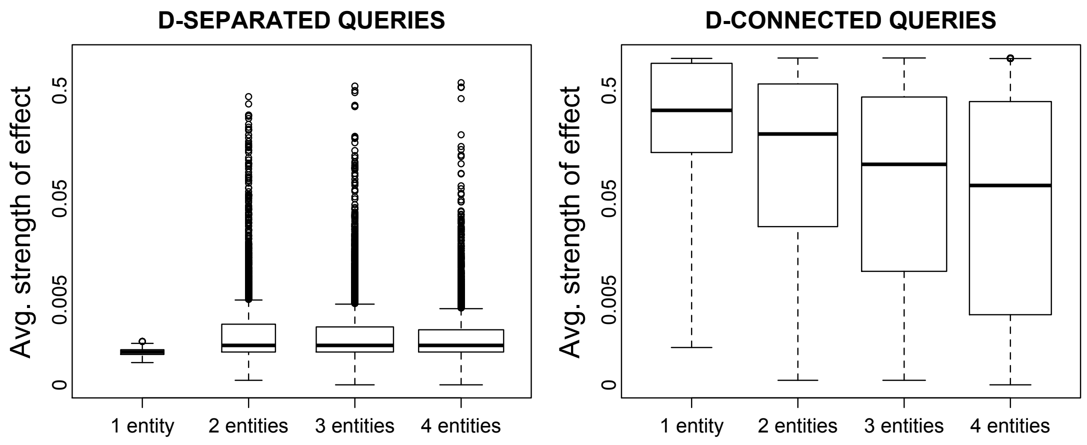


Figure 7.4: The average strength of effect of each query (measured as squared partial correlation) on actual data. (Left) Evaluating queries that the model claims to be d-separated or conditionally independent produces low average effect sizes. (Right) Queries that the model claims are d-connected or dependent produce high average effect sizes.

- The proportion of trials for which each query was deemed significant at $\alpha = 0.01$ adjusted using Bonferroni correction with the number of queries per trial

Figure 7.3 shows the distribution of the proportion of significant trials for both true (left) and false queries (right) for varying numbers of entities. Figure 7.4 shows the corresponding average strength of effects for true (left) and false (right) queries. The graph uses a standard box-and-whisker plot with values greater or less than 1.5 times the inner quartile range—the difference between the upper and lower quartiles—marked as outliers.

In the vast majority of cases, relational d-separation is consistent with tests on actual data (i.e., most d-separated queries have low effect sizes and are rarely deemed significant, whereas most d-connected queries have high effect sizes and are mostly deemed significant). For approximately 23,000 true queries, 14.9% are significant in more than one trial, but most are insubstantive, with only 2.2% having an average effect size greater than 0.01. There are three potential reasons why a d-separation in theory may appear to be d-connected in practice: (1) Type I error; (2) high power given a large sample size; or (3) bias. We have discovered that a small number of cases exhibit an interaction between aggregation and relational structure (i.e., degree or the cardinality of relational variable instances). This interaction violates the identically distributed assumption of data instances, which produces a biased estimate of effect size for simple linear regression. Linear regression does not account for these interaction effects, suggesting the need for more accurate statistical tests of conditional independence for relational data.

8. Model Assumptions and Related Work

The class of relational models considered in Section 4, while strictly more expressive than Bayesian networks, has limitations in its current formalization. In this section, we highlight these assumptions and discuss how related and future work could address them.

Self-relationships: Self-relationships are relationship classes that involve the same entity class more than once. Relational schemas, as defined in Definition 4.1, can express these types of relationships. Only the definition of relational paths—which govern the space of variables and dependencies—requires unique entity class names within [E,R,E] triples (see condition (2) of Definition 4.3). However, a common procedure in entity-relationship modeling is to map entity names to unique *role indicators* within the context of a self-relationship, such as manager/subordinate, friend1/friend2, or citing-paper/cited-paper (Ramakrishnan and Gehrke, 2002). This approach does not duplicate entity instances in the skeleton or ground graph; it only modifies their reference names within the relational path, requiring extended semantics for terminal sets. Incorporating self-relationships is a straightforward extension, but for simplicity, we omit this additional layer of complexity.

Relational autocorrelation: In contrast to self-relationships, relational autocorrelation is a statistical dependency among the values of the same attribute class frequently found in relational data sets (Jensen and Neville, 2002). Various models and learning algorithms have been developed to capture these types of dependencies, such as RDNs (Neville and Jensen, 2007), PBNs with an extended normal form (Schulte et al., 2012), and PRMs with dependencies that follow guaranteed acyclic relationships (Getoor et al., 2007). Our formalism, and equivalently PRMs (without guaranteed acyclic relationships), can represent a class of models for apparent autocorrelation. Any relational dependency that yields a common cause for grounded variables of the same attribute class—essentially any dependency that crosses a MANY cardinality—produces relational autocorrelation. The only autocorrelations not accounted for involve latent causes or those produced by temporal processes (e.g., feedback).

Context-specific independence: Context-specific independence (CSI) introduces independence of some variable and its parents, depending on the values of other variables. This can be achieved within the specification of conditional probability distributions as if-then-else statements of logical conditions, such as in DAPER models (Heckerman et al., 2007) or RPMs (Russell and Norvig, 2010), encoded as regularities in conditional probability tables (Boutilier et al., 1996), or with the recent graphical convention of gates (Minka and Winn, 2009). However, this introduces a notion of independence that cannot be inferred from model structure via traditional d-separation. In fact, Boutilier et al. (1996) define an analogous approach based on d-separation of a manipulated Bayesian network through deletion of vacuous dependencies given some context. Winn (2012) extends the rules of d-separation to reason over the additional paths and their collective state introduced by gates. An alternative and more general approach to encoding CSIs is to develop an *ontology* for which (in)dependencies hold depending on the type of entity or relationship. PRMs with class hierarchies allow a hierarchy of entity types where the dependency structure can vary depending on the type (Getoor et al., 2000). Rules of inheritance derived from object-oriented programming are used to define a coherent joint probability distribution. This aligns with our formalism, as relational schemas can be viewed as an ontology defined at a particular level. However, the semantics of d-separation under inheritance has not been developed and is a profitable direction of future research.

Causes of entity and relationship existence: Without a generative model of relational skeletons, the relational models are not truly generative as the skeleton must be generated prior to the attributes. However, the same issue occurs for Bayesian networks: Relational skeletons consist of disconnected entity instances, but the model does not specify how many instances to create. There are relational models that attempt to learn and represent models with unknown numbers of entity instances, such as Blog (Milch et al., 2005), or uncertain relationship instances, such as PRMs with existence uncertainty (Getoor et al., 2002). However, reasoning about the connection between conditional independence and existence is an open problem. For relationship existence, selection bias (conditioning) occurs when testing marginal dependence between variables across a particular relationship (Maier et al., 2010). For entity existence, some researchers argue that existence cannot be represented as a variable or predicate (Poole, 2007), while others represent them as predicates (Laskey, 2008). Therefore, we currently choose simple processes for generating skeletons, allowing us to focus on relational models of attributes and leaving structural causes and effects as future work.

Causal sufficiency: The relational models we consider assume that all common causes of observed variables are also observed and included in the model—an assumption commonly referred to as causal sufficiency. Many researchers have developed methods for learning and inference by explicitly modeling unobserved variables—typically termed latent variable models (Bishop, 1999)—or inferring the presence of latent entity classes—for example, latent group models (Neville and Jensen, 2005). However, only ancestral graphs and acyclic directed mixed graphs (ADMGs) do so in order to preserve an underlying conditional independence structure (Richardson and Spirtes, 2002; Richardson, 2009). These models are paired with the theory of m-separation, which is a generalization of d-separation for Bayesian networks. The generalization of ancestral graphs or ADMGs to relational models requires extensive theoretical exploration; therefore, we leave this as an important direction for future work. Given that a primary motivation for d-separation is to support constraint-based causal discovery, any relational extension to algorithms that learn causal models without assuming causal sufficiency, such as FCI (Spirtes et al., 1995; Zhang, 2008), its variants (Claassen and Heskes, 2011; Colombo et al., 2012), and BCCD (Claassen and Heskes, 2012), would require such an extension to m-separation.

Temporal and cyclic models: Currently, the relational model is assumed to be acyclic (with respect to the class dependency graph), and consequently, atemporal. Model-level cycles typically result from temporal processes for which grounding across time would yield an acyclic ground graph, such as in dynamic Bayesian networks (Dean and Kanazawa, 1989; Murphy, 2002). However, cycles can also be due to temporal processes where the interaction occurs at a faster rate than measurement. As a result, there has been considerable attention devoted to models that explicitly encode cyclic dependencies, such as the work by Spirtes (1995), Pearl and Dechter (1996), Richardson (1996), Dash (2005), Schmidt and Murphy (2009), and Hyttinen et al. (2012). Our formalism currently prohibits any relational dependency that has a common attribute class for the cause and effect, regardless of the relational path constraint. Relaxing this assumption would require either explicitly modeling temporal dynamics or enabling feedback loops. We reserve temporal dynamics and feedback as another important avenue for future research.

Despite these assumptions, our current work extends the notion of d-separation to a much more expressive class of models than Bayesian networks. This work is a first step toward deriving conditional independencies from expressive classes of models. Incorporating existence, ontologies, temporal dynamics and feedback, and latent variables into our model is important future work, especially in the context of representing and learning causal models of realistic domains.

9. Discussion

In this paper, we extend the theory of d-separation to graphical models of relational data. We present the *abstract ground graph*, a new representation that is sound and complete in its abstraction of dependencies across all possible ground graphs of a given relational model. We formally define relational d-separation and offer a sound, complete, and computationally efficient approach to deriving conditional independence facts from relational models by exploiting their abstract ground graphs. We also show that relational d-separation is equivalent to the Markov condition for relational models. We provide an empirical analysis of relational d-separation on synthetic data, demonstrating a close correspondence between the theory and statistical results in practice. Finally, we evaluate how frequently the additional complexity of abstract ground graphs proves necessary for accurately deriving conditional independence facts.

The results of this paper imply potential flaws in the design and analysis of some real-world studies. If researchers of social or economic systems choose inappropriate data and model representations, then their analyses may omit important classes of dependencies. Specifically, our theory implies that choosing a propositional representation from an inherently relational domain may lead to serious errors. An abstract ground graph from a given perspective defines the exact set of variables that must be included in any propositionalization. The absence of any relational variable (including intersection variables) may unnecessarily violate causal sufficiency, which could result in the inference of a causal dependency where conditional independence was not detected. Our work indicates that researchers should carefully consider how to represent their domains in order to accurately reason about conditional independence.

The abstract ground graph representation also presents an opportunity to derive new edge orientation rules for algorithms that learn the structure of relational models, such as RPC (Maier et al., 2010) and RCD (Maier et al., 2013). There are unique orientations of edges that are consistent with a given pattern of association that can only be recognized in an abstract ground graph. For example, in contrast to bivariate IID data, it is simple to establish the direction of causality for bivariate relational data. Consider the two bivariate, two-entity relational models depicted in Figure 9.1(a). The first model implies that values of X on A entities are caused by the values of Y on related B entities. The second model implies the opposite, that values of Y on B entities are caused by the values of X on related A entities. For simplicity, we show the relationship class only as a dashed line between entity classes and omit it from relational paths.

Figure 9.1(b) illustrates a fragment of the abstract ground graph (for hop threshold h=4) that each of the two relational models implies. As expected, the directions of the edges in the two abstract ground graphs are counterposed. Both models produce observable statistical dependencies for relational variable pairs $\langle [B].Y, [B, A].X\rangle$ and $\langle [B, A].X, [B, A, B].Y\rangle$. However, the relational variables $[B].Y$ and $[B, A, B].Y$ have different observable statistical dependencies: In the first model, they are marginally independent and conditionally dependent given $[B, A].X$, and in the second model, they are marginally dependent and

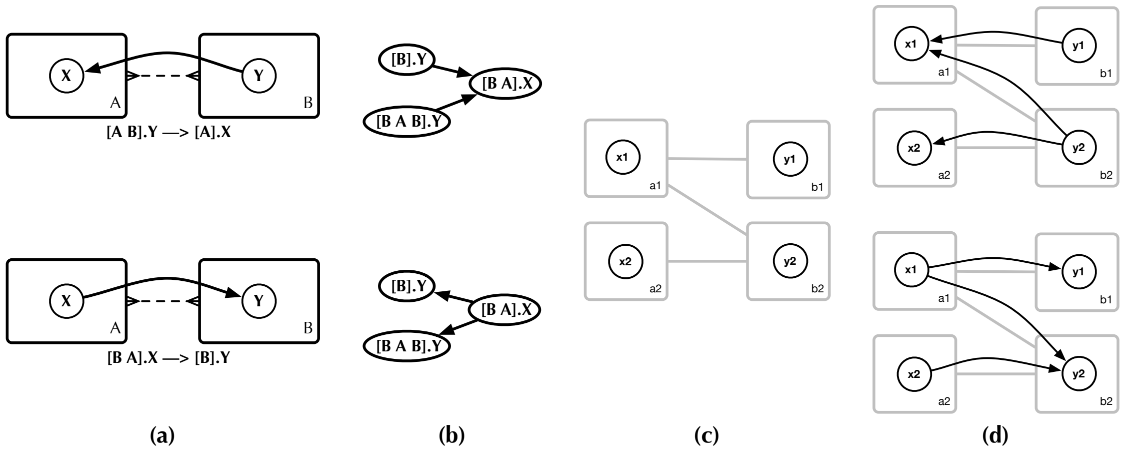


Figure 9.1: (a) Two models of a bivariate relational domain with opposite directions of causality for a single dependency (relationship class omitted for simplicity); (b) a single dependency implies additional dependencies among arbitrary relational variables, shown here in a fragment of the abstract ground graph for B's perspective; (c) an example relational skeleton; and (d) the ground graphs resulting from applying the relational model to the skeleton.

conditionally independent given $[B, A].X$. As a result, we can uniquely determine the direction of causality of the single dependence by exploiting relational structure. (There is symmetric reasoning for relational variables from A's perspective, and this result is also applicable to ONE-to-MANY data.)

To illustrate this fact more concretely, consider the small relational skeleton shown in Figure 9.1(c) and the ground graphs applied to this skeleton in Figure 9.1(d). In the first ground graph, we have $y_1 \perp\!\!\!\perp y_2$ and $y_1 \not\perp\!\!\!\perp y_2 \mid x_1$, but in the second ground graph, we have $y_1 \not\perp\!\!\!\perp y_2$ and $y_1 \perp\!\!\!\perp y_2 \mid x_1$. These opposing conditional independence relations uniquely determine the correct causal model. In prior work, we formalized this idea as a new rule, called relational bivariate orientation (RBO) (Maier et al., 2013), to orient dependencies in a constraint-based causal discovery algorithm.

Deriving and formalizing the implications of relational d-separation is a main direction of future research. Additionally, our experiments suggest that more accurate tests of conditional independence for relational data need to be developed, specifically tests that can address the interaction between relational structure and aggregation across terminal sets of relational variables. This work has also focused solely on relational models of attributes; future work should consider models of relationship and entity existence to fully characterize generative models of relational structure. The theory could also be extended to incorporate functional or deterministic dependencies, as D-separation extends d-separation for Bayesian networks. Finally, the work on identifying causal effects in Bayesian networks could be extended to relational models. This may similarly require an extension of do-calculus to consider the space of relational interventions, which may include adding or removing entity or relationship instances, as well as fixing attribute values.

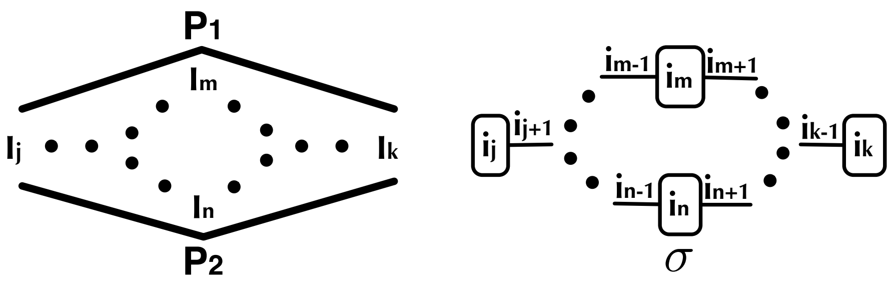


Figure A.1: Schematic of two relational paths P_1 and P_2 for which Lemma 4.1 guarantees that some skeleton σ yields a non-empty intersection of their terminal sets. The example depicts a possible constructed skeleton based on the procedure used in the proof of Lemma 4.1.

Acknowledgments

The authors wish to thank Cindy Loiselle for her editing expertise. The authors also thank the anonymous reviewers for their helpful comments, prompting us to create a much more readable, precise, correct, and useful paper. This effort is supported by the Intelligence Advanced Research Project Agency (IARPA) via Department of Interior National Business Center Contract number D11PC20152, Air Force Research Lab under agreement number FA8750-09-2-0187, the National Science Foundation under grant number 0964094, and Science Applications International Corporation (SAIC) and DARPA under contract number P010089628. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoI/NBC, AFRL, NSF, SAIC, DARPA, or the U.S. Government. Katerina Marazopoulou received scholarship support from the Greek State Scholarships Foundation.

Appendix A. Proofs

In this appendix, we provide detailed proofs for all previous lemmas, theorems, and corollaries.

Lemma 4.1 *For two relational paths of arbitrary length from I_j to I_k that differ in at least one item class, $P_1 = [I_j, \ldots, I_m, \ldots, I_k]$ and $P_2 = [I_j, \ldots, I_n, \ldots, I_k]$ with $I_m \neq I_n$, there exists a skeleton $\sigma \in \Sigma_{\mathcal{S}}$ such that $P_1|_{i_j} \cap P_2|_{i_j} \neq \emptyset$ for some $i_j \in \sigma(I_j)$.*

Proof. Proof by construction. Let $\mathcal{S}$ be an arbitrary schema with two arbitrary relational paths $P_1 = [I_j, \ldots, I_m, \ldots, I_k]$ and $P_2 = [I_j, \ldots, I_n, \ldots, I_k]$ where $I_m \neq I_n$. We will construct a skeleton $\sigma \in \Sigma_{\mathcal{S}}$ such that the terminal sets for item $i_j \in \sigma(I_j)$ along P_1 and P_2 have a non-empty intersection, that is, an item $i_k \in P_1|_{i_j} \cap P_2|_{i_j} \neq \emptyset$ (roughly depicted in Figure A.1). We use the following procedure to build σ:

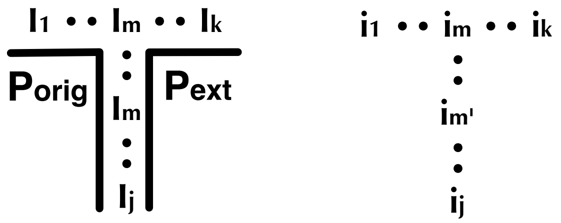


Figure A.2: Example construction of a relational skeleton for two relational paths $P_{orig} = [I_1, \ldots, I_m \ldots, I_j]$ and $P_{ext} = [I_j, \ldots, I_m \ldots, I_k]$, where item class I_m is repeated between I_m and I_j. This construction is used within the proof of Lemma 5.1.

1. Simultaneously traverse P_1 and P_2 from I_j until the paths diverge. For each entity class $E \in \mathcal{E}$ reached, add a unique entity instance e to $\sigma(E)$.
2. Simultaneously traverse P_1 and P_2 backwards from I_k until the paths diverge. For each entity class $E \in \mathcal{E}$ reached, add a unique entity instance e to $\sigma(E)$.
3. For the divergent subpaths of both P_1 and P_2, add unique entity instances for each entity class $E \in \mathcal{E}$.
4. Repeat 1–3 for relationship classes. For each $R \in \mathcal{R}$ reached, add a unique relationship instance r connecting the entity instances from classes on P_1 and P_2, and add unique entity instances for classes $E \in R$ not appearing on P_1 and P_2.

This process constructs an admissible skeleton—all instances are unique and this process assumes no cardinality constraints aside from those required by Definition 4.3. By construction, there exists an item $i_j \in \sigma(I_j)$ such that $P_1|_{i_j} \cap P_2|_{i_j} = \{i_k\} \neq \emptyset$. ■

Lemma 5.1 *Let $P_{orig} = [I_1, \ldots, I_j]$ and $P_{ext} = [I_j, \ldots, I_k]$ be two relational paths with $\mathbf{P} = extend(P_{orig}, P_{ext})$. Then, $\forall P \in \mathbf{P}$ there exists a relational skeleton $\sigma \in \Sigma_{\mathcal{S}}$ such that $\exists i_1 \in \sigma(I_1)$ such that $\exists i_k \in P|_{i_1}$ and $\exists i_j \in P_{orig}|_{i_1}$ such that $i_k \in P_{ext}|_{i_j}$.*

Proof. Let $P \in \mathbf{P}$ be an arbitrary valid relational path, where $P = P_{orig}^{1,n_o-c+1} + P_{ext}^{c+1,n_e}$ for pivot c. There are two subcases:

(a) $c = 1$ and $P = [I_1, \ldots, I_j, \ldots, I_k]$. This subcase holds generally for any skeleton. Proof by contradiction. Let σ be an arbitrary skeleton, choose $i_1 \in \sigma(I_1)$ arbitrarily, and choose $i_k \in P|_{i_1}$ arbitrarily. Assume for contradiction that there is no i_j in the terminal set $P_{orig}|_{i_1}$ such that i_k would be in the terminal set $P_{ext}|_{i_j}$, that is, $\forall i_j \in P_{orig}|_{i_1}$ $i_k \notin P_{ext}|_{i_j}$. Since $P = [I_1, \ldots, I_j, \ldots, I_k]$, we know that i_k is reached by traversing σ from i_1 via some i_j to i_k. But the traversal from i_1 to i_j implies that $i_j \in [I_1, \ldots, I_j]|_{i_1} = P_{orig}|_{i_1}$, and the traversal from i_j to i_k implies that $i_k \in [I_j, \ldots, I_k]|_{i_j} = P_{ext}|_{i_j}$. Therefore, there must exist an $i_j \in P_{orig}|_{i_1}$ such that $i_k \in P_{ext}|_{i_j}$.

(b) $c > 1$ and $P = [I_1, \ldots, I_m, \ldots, I_k]$. Proof by construction. We build a relational skeleton σ following the same procedure as outlined in the proof of Lemma 4.1. Add instances to σ for every item class that appears on P_{orig} and P_{ext}. Since $P = [I_1, \ldots, I_m, \ldots, I_k]$, we know that i_k is reached by traversing σ from i_1 via some i_m to i_k. By case (a), $\exists i_m \in [I_1, \ldots, I_m]|_{i_1}$ such that $i_k \in [I_m, \ldots, I_k]|_{i_m}$. We then must show that

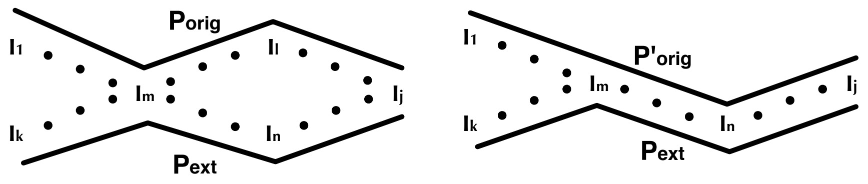


Figure A.3: Schematic of the relational paths expected in Lemma 5.2. If item i_k is unreachable via $extend(P_{orig}, P_{ext})$, then there must exist a P'_{orig} of the form $[I_1, \ldots, I_m, \ldots, I_n, \ldots, I_j]$.

there exists an $i_j \in [I_m, \ldots, I_j]|_{i_m}$ with $i_m \in [I_j, \ldots, I_m]|_{i_j}$. But constructing the skeleton with unique item instances for every appearance of an item class on the relational paths provides this and does not violate any cardinality constraints. If any item class appears more than once, then the bridge burning semantics are induced. However, adding an additional item instance for every reappearance of an item class enables the traversal from i_j to i_m and vice versa. An example of this construction is displayed in Figure A.2. This is also a valid relational skeleton because P_{orig} and P_{ext} are valid relational paths, and by definition, the cardinality constraints of the schema permit multiple instances in the skeleton of any repeated item class. By this procedure, we show that there exists a skeleton σ such that there exists an $i_j \in P_{orig}|_{i_1}$ such that $i_k \in P_{ext}|_{i_j}$. ■

Lemma 5.2 *Let $\sigma \in \Sigma_{\mathcal{S}}$ be a relational skeleton, and let $P_{orig} = [I_1, \ldots, I_j]$ and $P_{ext} = [I_j, \ldots, I_k]$ be two relational paths with $\mathbf{P} = extend(P_{orig}, P_{ext})$. Then, $\forall i_1 \in \sigma(I_1)$ $\forall i_j \in P_{orig}|_{i_1}$ $\forall i_k \in P_{ext}|_{i_j}$ if $\forall P \in \mathbf{P}$ $i_k \notin P|_{i_1}$, then $\exists P'_{orig}$ such that $P_{orig}|_{i_1} \cap P'_{orig}|_{i_1} \neq \emptyset$ and $i_k \in P'|_{i_1}$ for some $P' \in extend(P'_{orig}, P_{ext})$.*

Proof. Proof by construction. Let $i_1 \in \sigma(I_1)$, $i_j \in P_{orig}|_{i_1}$, and $i_k \in P_{ext}|_{i_j}$ be arbitrary instances such that $i_k \notin P|_{i_1}$ for all $P \in \mathbf{P}$.

Since $i_j \in P_{orig}|_{i_1}$ and $i_k \in P_{ext}|_{i_j}$, but $i_k \notin P|_{i_1}$, there exists no pivot that yields a common subsequence in P_{orig} and P_{ext} that produces a path in *extend* that can reach i_k. Let the first divergent item class along the reverse of P_{orig} be I_l and along P_{ext} be I_n. The two paths must not only diverge, but they also necessarily reconverge at least once. If P_{orig} and P_{ext} do not reconverge, then there are no reoccurrences of an item class along any $P \in \mathbf{P}$ that would restrict the inclusion of i_k in some terminal set $P|_{i_1}$. The sole reason that $i_k \notin P|_{i_1}$ for all $P \in \mathbf{P}$ is due to the bridge burning semantics specified in Definition 4.4.

Without loss of generality, assume P_{orig} and P_{ext} reconverge once, at item class I_m. So, $P_{orig} = [I_1, \ldots, I_m, \ldots, I_l, \ldots, I_j]$ and $P_{ext} = [I_j, \ldots, I_n, \ldots, I_m, \ldots, I_k]$ with $I_l \neq I_n$, as depicted in Figure A.3. Let $P'_{orig} = [I_1, \ldots, I_m, \ldots, I_n, \ldots, I_j]$, which is a valid relational path because $[I_1, \ldots, I_m]$ is a subpath of P_{orig} and $[I_m, \ldots, I_n, \ldots, I_j]$ is a subpath of P_{ext}.

By construction, $i_j \in P_{orig}|_{i_1} \cap P'_{orig}|_{i_1}$. If $P' = [I_1, \ldots, I_m, \ldots, I_k] \in extend(P'_{orig}, P_{ext})$ with pivot at I_m, then $i_k \in P'|_{i_1}$. ■

Theorem 5.2 *For every acyclic relational model structure $\mathcal{M}$ and perspective $B \in \mathcal{E} \cup \mathcal{R}$, the abstract ground graph $AGG_{\mathcal{M}B}$ is sound and complete for all ground graphs $GG_{\mathcal{M}\sigma}$ with skeleton $\sigma \in \Sigma_{\mathcal{S}}$.*

Proof. Let $\mathcal{M} = (\mathcal{S}, \mathcal{D})$ be an arbitrary acyclic relational model structure and let $B \in \mathcal{E} \cup \mathcal{R}$ be an arbitrary perspective.

Soundness: To prove that $AGG_{\mathcal{M}B}$ is sound, we must show that for every edge $P_k.X \to P_j.Y$ in $AGG_{\mathcal{M}B}$, there exists a corresponding edge $i_k.X \to i_j.Y$ in the ground graph $GG_{\mathcal{M}\sigma}$ for some skeleton $\sigma \in \Sigma_{\mathcal{S}}$, where $i_k \in P_k|_b$ and $i_j \in P_j|_b$ for some $b \in \sigma(B)$. There are three subcases, one for each type of edge in an abstract ground graph:

(a) Let $[B, \ldots, I_k].X \to [B, \ldots, I_j].Y \in \mathit{RVE}$ be an arbitrary edge in $AGG_{\mathcal{M}B}$ between a pair of relational variables. Assume for contradiction that there exists no edge $i_k.X \to i_j.Y$ in any ground graph:

$$\forall \sigma \in \Sigma_{\mathcal{S}}\ \forall b \in \sigma(B)\ \forall i_k \in [B, \ldots, I_k]|_b\ \forall i_j \in [B, \ldots, I_j]|_b\ \big(i_k.X \to i_j.Y \notin GG_{\mathcal{M}\sigma}\big)$$

By Definition 5.2 for abstract ground graphs, if $[B, \ldots, I_k].X \to [B, \ldots, I_j].Y \in \mathit{RVE}$, then the model must have dependency $[I_j, \ldots, I_k].X \to [I_j].Y \in \mathcal{D}$ such that $[B, \ldots, I_k] \in \mathit{extend}([B, \ldots, I_j], [I_j, \ldots, I_k])$. So, by Definition 4.9 for ground graphs, there is an edge from every $i_k.X$ to every $i_j.Y$, where i_k is in the terminal set for i_j along $[I_j, \ldots, I_k]$:

$$\forall \sigma \in \Sigma_{\mathcal{S}}\ \forall i_j \in \sigma(I_j)\ \forall i_k \in [I_j, \ldots, I_k]|_{i_j}\ \big(i_k.X \to i_j.Y \in GG_{\mathcal{M}\sigma}\big)$$

Since $[B, \ldots, I_k] \in \mathit{extend}([B, \ldots, I_j], [I_j, \ldots, I_k])$, by Lemma 5.1 we know that

$$\exists \sigma \in \Sigma_{\mathcal{S}}\ \exists b \in \sigma(B)\ \exists i_k \in [B, \ldots, I_k]|_b\ \exists i_j \in [B, \ldots, I_j]|_b\ \big(i_k \in [I_j, \ldots, I_k]|_{i_j}\big)$$

Therefore, there exists a ground graph $GG_{\mathcal{M}\sigma}$ such that $i_k.X \to i_j.Y \in GG_{\mathcal{M}\sigma}$, which contradicts the assumption.

(b) Let $P_1.X \cap P_2.X \to [B, \ldots, I_j].Y \in \mathit{IVE}$ be an arbitrary edge in $AGG_{\mathcal{M}B}$ between an intersection variable and a relational variable, where $P_1 = [B, \ldots, I_m, \ldots, I_k]$ and $P_2 = [B, \ldots, I_n, \ldots, I_k]$ with $I_m \neq I_n$. By Lemma 4.1, there exists a skeleton $\sigma \in \Sigma_{\mathcal{S}}$ and $b \in \sigma(B)$ such that $P_1|_b \cap P_2|_b \neq \emptyset$. Let $i_k \in P_1|_b \cap P_2|_b$ and assume for contradiction that for all $i_j \in [B, \ldots, I_j]|_b$ there is no edge $i_k.X \to i_j.Y$ in the ground graph $GG_{\mathcal{M}\sigma}$. By Definition 5.2, if the abstract ground graph has edge $P_1.X \cap P_2.X \to [B, \ldots, I_j].Y \in \mathit{IVE}$, then either $P_1.X \to [B, \ldots, I_j].Y \in \mathit{RVE}$ or $P_2.X \to [B, \ldots, I_j].Y \in \mathit{RVE}$. Then, as shown in case (a), there exists an $i_j \in [B, \ldots, I_j]|_b$ such that $i_k.X \to i_j.Y \in GG_{\mathcal{M}\sigma}$, which contradicts the assumption.

(c) Let $[B, \ldots, I_k].X \to P_1.Y \cap P_2.Y \in \mathit{IVE}$ be an arbitrary edge in $AGG_{\mathcal{M}B}$ between a relational variable and an intersection variable, where $P_1 = [B, \ldots, I_m, \ldots, I_j]$ and $P_2 = [B, \ldots, I_n, \ldots, I_j]$ with $I_m \neq I_n$. The proof follows case (b) to show that there exists a skeleton $\sigma \in \Sigma_{\mathcal{S}}$ and $b \in \sigma(B)$ such that for all $i_k \in [B, \ldots, I_k]|_b$ there exists an $i_j \in P_1 \cap P_2|_b$ such that $i_k.X \to i_j.Y \in GG_{\mathcal{M}\sigma}$.

Completeness: To prove that the abstract ground graph $AGG_{\mathcal{M}B}$ is complete, we show that for every edge $i_k.X \to i_j.Y$ in every ground graph $GG_{\mathcal{M}\sigma}$ where $\sigma \in \Sigma_{\mathcal{S}}$, there is a set of corresponding edges in $AGG_{\mathcal{M}B}$. Specifically, the edge $i_k.X \to i_j.Y$ yields two sets of relational variables for some $b \in \sigma(B)$, namely $\mathbf{P_k.X} = \{P_k.X \mid i_k \in P_k|_b\}$ and

$\mathbf{P_j.Y} = \{P_j.Y \mid i_j \in P_j|_b\}$. Note that all relational variables in both $\mathbf{P_k.X}$ and $\mathbf{P_j.Y}$ are nodes in $AGG_{\mathcal{M}B}$, as are all pairwise intersection variables: $\forall P_k.X, P'_k.X \in \mathbf{P_k.X}\ \big(P_k.X \cap P'_k.X \in AGG_{\mathcal{M}B}\big)$ and $\forall P_j.Y, P'_j.Y \in \mathbf{P_j.Y}\ \big(P_j.Y \cap P'_j.Y \in AGG_{\mathcal{M}B}\big)$. We show that for all $P_k.X \in \mathbf{P_k.X}$ and for all $P_j.Y \in \mathbf{P_j.Y}$ either (a) $P_k.X \to P_j.Y \in AGG_{\mathcal{M}B}$, (b) $P_k.X \cap P'_k.X \to P_j.Y \in AGG_{\mathcal{M}B}$, where $P'_k.X \in \mathbf{P_k.X}$, or (c) $P_k.X \to P_j.Y \cap P'_j.Y \in AGG_{\mathcal{M}B}$, where $P'_j.Y \in \mathbf{P_j.Y}$.

Let $\sigma \in \Sigma_{\mathcal{S}}$ be an arbitrary skeleton, let $i_k.X \to i_j.Y \in GG_{\mathcal{M}\sigma}$ be an arbitrary edge drawn from $[I_j, \ldots, I_k].X \to [I_j].Y \in \mathcal{D}$, and let $P_k.X \in \mathbf{P_k.X}, P_j.Y \in \mathbf{P_j.Y}$ be an arbitrary pair of relational variables.

(a) If $P_k \in \mathit{extend}(P_j, [I_j, \ldots, I_k])$, then $P_k.X \to P_j.Y \in AGG_{\mathcal{M}B}$ by Definition 5.2.

(b) If $P_k \notin \mathit{extend}(P_j, [I_j, \ldots, I_k])$, but $\exists P'_k \in \mathit{extend}(P_j, [I_j, \ldots, I_k])$ such that $P'_k.X \in \mathbf{P_k.X}$, then $P'_k.X \to P_j.Y \in AGG_{\mathcal{M}B}$, and $P_k.X \cap P'_k.X \to P_j.Y \in AGG_{\mathcal{M}B}$ by Definition 5.2.

(c) If $\forall P \in \mathit{extend}(P_j, [I_j, \ldots, I_k])\ \big(P.X \notin \mathbf{P_k.X}\big)$, then by Lemma 5.2, $\exists P'_j$ such that $i_j \in P'_j|_b$ and $P_k \in \mathit{extend}(P'_j, [I_j, \ldots, I_k])$. Therefore, $P'_j.Y \in \mathbf{P_j.Y}$, $P_k.X \to P'_j.Y \in AGG_{\mathcal{M}B}$, and $P_k.X \to P'_j.Y \cap P_j.Y \in AGG_{\mathcal{M}B}$ by Definition 5.2. ■

Theorem 5.3 *For every acyclic relational model structure $\mathcal{M}$ and perspective $B \in \mathcal{E} \cup \mathcal{R}$, the abstract ground graph $AGG_{\mathcal{M}B}$ is directed and acyclic.*

Proof. Let $\mathcal{M}$ be an arbitrary acyclic relational model structure, and let $B \in \mathcal{E} \cup \mathcal{R}$ be an arbitrary perspective. It is clear by Definition 5.2 that every edge in the abstract ground graph $AGG_{\mathcal{M}B}$ is directed by construction. All edges inserted in any abstract ground graph are drawn from the directed dependencies in a relational model. Since $\mathcal{M}$ is acyclic, the class dependency graph $G_{\mathcal{M}}$ is also acyclic by Definition 4.10. Assume for contradiction that there exists a cycle of length n in $AGG_{\mathcal{M}B}$ that contains both relational variables and intersection variables. By Definition 5.2, all edges inserted in $AGG_{\mathcal{M}B}$ are drawn from some dependency in $\mathcal{M}$, even for nodes corresponding to intersection variables. Retaining only the final item class in each relational path for every node in the cycle will yield a cycle in $G_{\mathcal{M}}$ by Definition 4.10. Therefore, $\mathcal{M}$ could not have been acyclic, which contradicts the assumption. ■

Appendix B. The Semantics of Bridge Burning

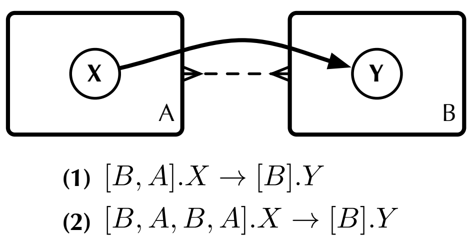
In this appendix, we provide an example to show that the bridge burning semantics for terminal sets of relational paths yields a strictly more expressive class of relational models than semantics without bridge burning. The bridge burning semantics produces terminal sets that are necessarily *subsets* of terminal sets which would otherwise be produced without bridge burning. Paradoxically, this enables a *superset* of relational models.

Recall the definition of a terminal set for a relational path:

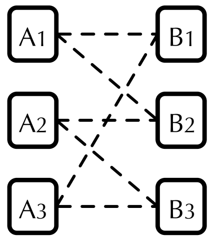
Definition 4.4 (Terminal set) For skeleton $\sigma \in \Sigma_{\mathcal{S}}$ and $i_j \in \sigma(I_j)$, the *terminal set* $P|_{i_j}$ for relational path $P = [I_j, \ldots, I_k]$ of length n is defined inductively as

$$P^1|_{i_j} = [I_j]|_{i_j} = \{i_j\}$$

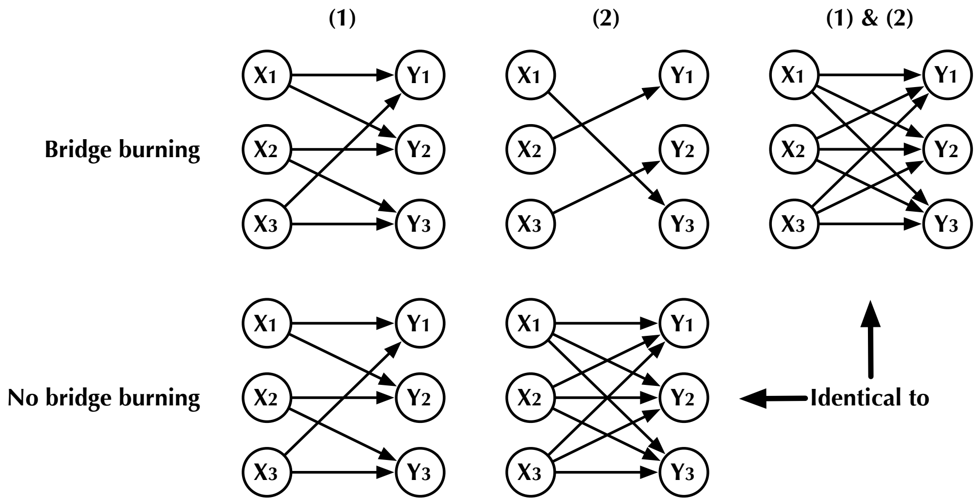
$$\vdots$$



(a) Relational model



(b) Relational skeleton



(c) Ground graphs

Figure B.1: Example demonstrating that bridge burning semantics yields a more expressive class of models than semantics without bridge burning. (a) Relational model over a schema with two entity classes and two attributes with two possible relational dependencies (relationship class omitted for simplicity). (b) Simple relational skeleton with three A and three B instances. (c) Bridge burning semantics yields three possible ground graphs with combinations of dependencies (1) and (2), whereas no bridge burning yields two possible ground graphs. The bridge burning ground graphs subsume the ground graphs without bridge burning.

$$P^n|_{i_j} = [I_j, \ldots, I_k]|_{i_j} = \bigcup_{i_m \in P^{n-1}|_{i_j}} \Big\{ i_k \mid \big((i_m \in i_k \text{ if } I_k \in \mathcal{R}) \ \vee \ (i_k \in i_m \text{ if } I_k \in \mathcal{E})\big) \wedge i_k \notin \bigcup_{l=1}^{n-1} P^l|_{i_j} \Big\}$$

The final condition in the inductive definition ($i_k \notin [I_1, \ldots, I_j]|_{i_1}$ for $j = 1$ to $k-1$) encodes bridge burning. The item i_k is only added to the terminal set if it is not a member of the terminal set of any previous subpath. For example, let P be the relational path [Employee, Develops, Product, Develops, Employee]. This relational path produces terminal sets that include the employees that work on the same products (that is, co-workers). Instantiating this path with the employee Quinn, $P|_{\text{Quinn}}$, produces the terminal set {Paul, Roger, Sally}. Since Quinn $\in$ [Employee]$|_{\text{Quinn}}$, the bridge burning semantics excludes Quinn from this set. This makes intuitive sense as well—Quinn should not be considered her own colleague.

A relational model is simply a collection of relational dependencies. Each relational dependency is primarily described by the relational path of the parent relational variable (because, for canonically specified dependencies, the relational path of the child consists of a single item class). The relational path specification is used in the construction of ground graphs, connecting variable instances that appear in the terminal sets of the parent and child relational variables.

To characterize the expressiveness of relational models, we can inspect the space of representable ground graphs by choosing an arbitrary relational skeleton and a small set of relational dependencies. We show with a simple example that the bridge burning semantics for a model over a two-entity, bivariate schema yields more possible ground graphs than without bridge burning. (We omit the relationship class for simplicity.) In Figure B.1(a), we present such a model with two possible relational dependencies labeled (1) and (2). Figure B.1(b) provides a simple relational skeleton involving three A and three B instances (relationship instances are represented as dashed lines for simplicity). As shown in Figure B.1(c), the bridge burning semantics leads to three possible ground graphs, one for each combination of the dependencies (1), (2), and both (1) and (2) together. Without bridge burning, only two ground graphs are possible because dependency (2) completely subsumes dependency (1) with those semantics.

This example generalizes to arbitrary dependencies. The terminal sets of relational paths that repeat item classes subsume subpaths under the semantics without bridge burning. This leads to fewer possible relational models, which justifies our choice of semantics for terminal sets of relational paths.

Appendix C. Soundness and Completeness of Relational Paths

In this appendix, we prove that the definition of relational paths (repeated below) is sound and complete with respect to producing non-empty terminal sets for at least one relational skeleton.

Definition 4.3 (Relational path) A *relational path* $[I_j, \ldots, I_k]$ for relational schema $\mathcal{S}$ is an alternating sequence of entity and relationship classes $I_j, \ldots, I_k \in \mathcal{E} \cup \mathcal{R}$ such that:

(1) For every pair of consecutive item classes $[E, R]$ or $[R, E]$ in the path, $E \in R$.
(2) For every triple of consecutive item classes $[E, R, E']$, $E \neq E'$.
(3) For every triple of consecutive item classes $[R, E, R']$, if $R = R'$, then $card(R, E) =$ many.

Lemma C.1 *Let $\mathcal{S}$ be a relational schema and $[I_j, \ldots, I_k]$ be a sequence of alternating entity and relationship classes of $\mathcal{S}$ that satisfy participation constraints (condition (1) of Definition 4.3). The relational path $[I_j, \ldots, I_k]$ satisfies conditions (2) and (3) of Definition 4.3 if and only if there exists a relational skeleton $\sigma \in \Sigma_{\mathcal{S}}$ and an item instance $i_j \in \sigma(I_j)$ such that $[I_j, \ldots, I_k]|_{i_j} \neq \emptyset$. More formally,*

$$\exists \sigma \in \Sigma_{\mathcal{S}}\ \exists i_j \in \sigma(I_j)\ \big([I_j, \ldots, I_k]|_{i_j} \neq \emptyset\big) \Leftrightarrow \begin{array}{l} \big([ERE]\ \notin [I_j, \ldots, I_k]\big) \\ \qquad\qquad \wedge \\ \big([RER]\ \in [I_j, \ldots, I_k] \rightarrow \mathit{card}(R, E) = \text{MANY}\big) \end{array}$$

Proof. Left-to-right $\Rightarrow$: Assume that there exists a skeleton $\sigma \in \Sigma_{\mathcal{S}}$ and item instance $i_j \in \sigma(I_j)$ such that $[I_j, \ldots, I_k]|_{i_j} \neq \emptyset$. We must show that $[I_j, \ldots, I_k]$ obeys conditions (2) and (3), i.e., $[I_j, \ldots, I_k]$ does not contain any $[ERE]$ patterns, and if it contains an $[RER]$ pattern, then $\mathit{card}(R, E) = \text{MANY}$.

- Assume for contradiction that $[I_j, \ldots, I_k]$ contains a pattern of the form $[ERE]$. From Definition 4.4 for terminal sets, it follows that if the terminal set of a path is not empty, then the terminal set of every prefix of that path is not empty:

  $$[I_j, \ldots, I_k]|_{i_j} \neq \emptyset \Rightarrow [I_j, \ldots, I_m]|_{i_j} \neq \emptyset \text{ for all } [I_j, \ldots, I_m] \leq [I_j, \ldots, I_k]$$

  By assumption, $[I_j, \ldots, I_k]|_{i_j} \neq \emptyset$; therefore, the prefix $[I_j, \ldots, I_m]$ that ends in the ERE pattern also has a non-empty terminal set:

  $$\begin{aligned} [I_j, \ldots, I_k]|_{i_j} \neq \emptyset &\Rightarrow [I_j, \ldots, E, R, E]|_{i_j} \neq \emptyset \\ [I_j, \ldots, I_k]|_{i_j} \neq \emptyset &\Rightarrow [I_j, \ldots, E, R]|_{i_j} \neq \emptyset \\ [I_j, \ldots, I_k]|_{i_j} \neq \emptyset &\Rightarrow [I_j, \ldots, E]|_{i_j} \neq \emptyset \end{aligned}$$

  Let $e \in \sigma(E)$ be an entity instance in the terminal set $[I_j, \ldots, E]|_{i_j}$. Since the terminal set $[I_j, \ldots, E, R]|_{i_j}$ is not empty, it follows that there exists a relationship instance $r = \langle \ldots, e, \ldots \rangle$ such that $r \in [I_j, \ldots, E, R]|_{i_j}$. However, $[I_j, \ldots, E, R, E]|_{i_j}$ is also not empty; thus, there exists some $e' \in \sigma(E)$ such that $e' \in [I_j, \ldots, E, R, E]|_{i_j}$, where $e' \neq e$, and $e' \in r$. It follows that both e and e' participate in the relationship instance r, which is a contradiction.

- Assume for contradiction that $[I_j, \ldots, I_k]$ contains a pattern of the form $[R, E, R]$ and $\mathit{card}(R, E) = \text{ONE}$.

  $$\begin{aligned} [I_j, \ldots, R]|_{i_j} \neq \emptyset &\Rightarrow \exists r = \langle e, \ldots \rangle \in [I_j, \ldots, R]|_{i_j} && (1) \\ [I_j, \ldots, R, E]|_{i_j} \neq \emptyset &\Rightarrow \exists e \in [I_j, \ldots, R, E]|_{i_j} \text{ and } e \in r \\ [I_j, \ldots, R, E, R]|_{i_j} \neq \emptyset &\Rightarrow \exists r' = \langle e, \ldots \rangle \text{ such that } r' \in [I_j, \ldots, R, E, R]|_{i_j} && (2) \\ &\qquad\qquad \text{and } r' \neq r \text{ (bridge burning semantics)} \end{aligned}$$

  From (1) and (2) it follows that e participates in two instances of R; therefore, $\mathit{card}(R, E)$ must be MANY, which is a contradiction.

Right-to-left $\Leftarrow$: Assume that $[I_j, \ldots, I_k]$ adheres to Definition 4.3 for relational paths. We must show that $\exists \sigma \in \Sigma_{\mathcal{S}}\ \exists i_j \in \sigma(I_j)\ \big([I_j, \ldots, I_k]|_{i_j} \neq \emptyset\big)$. We can construct such a skeleton σ according to the following procedure: For each entity class E on the path, add a unique entity instance e to $\sigma(E)$. Then, for each relationship class R on the path, add a unique relationship instance r connecting the previously created unique entity instances that participate in R, and add unique entity instances for classes $E \in R$ not appearing on the path. This process constructs an admissible skeleton—all instances are unique and this process assumes no cardinality constraints aside from those required by Definition 4.3. By construction, there exists an item instance $i_j \in \sigma(I_j)$ such that $[I_j, \ldots, I_k]|_{i_j} \neq \emptyset$. ■

Appendix D. Background on Propositional Data and Models

In this appendix, we provide a brief review of Bayesian networks, traditional d-separation, and their connection to causality. We also describe why the class of Bayesian networks is a special case of relational models. Finally, we give an example of how to propositionalize a data set drawn from a relational domain.

A common assumption in classical statistics, machine learning, and causal discovery is that data instances are independent and identically distributed (IID). The first condition assumes that the variables on any given data instance are marginally independent of the variables of any other data instance. The second condition assumes that every data instance is drawn from the same underlying joint probability distribution. IID data (also referred to as propositional data[12]) are effectively represented as a single table, where rows correspond to the independent instances and columns are attributes of those instances.

A Bayesian network is a widely used probabilistic graphical model of propositional data (Pearl, 1988). A Bayesian network is represented as a directed acyclic graph $G = (\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is a set of vertices corresponding to random variables in the data and $\mathbf{E} \subset \mathbf{V} \times \mathbf{V}$ is a set of edges encoding the probabilistic dependencies among the variables. Each random variable $V \in \mathbf{V}$ is associated with a conditional probability distribution $P\big(V \mid \mathit{parents}(V)\big)$, where $\mathit{parents}(V) \subseteq \mathbf{V} \setminus \{V\}$ is the set of parent variables for V.

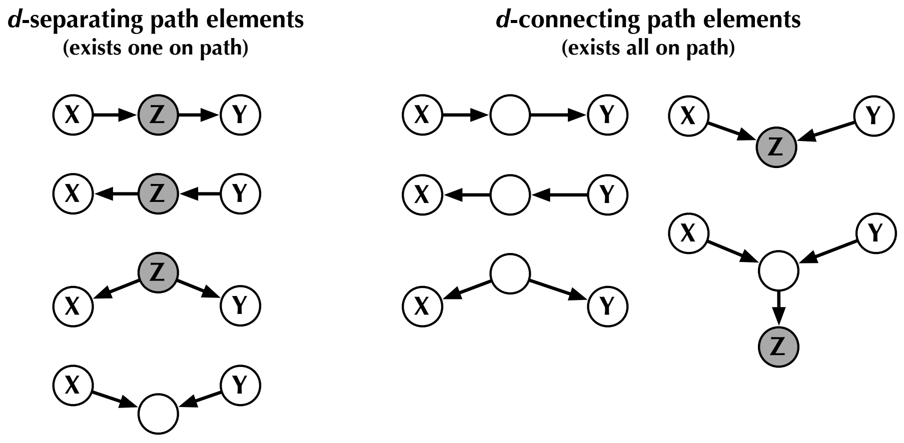
If the joint probability distribution $P(\mathbf{V})$ satisfies the Markov condition for G, then $P(\mathbf{V})$ can be factored as $\prod_{V \in \mathbf{V}} P\big(V \mid \mathit{parents}(V)\big)$ using the conditional distributions. The Markov condition states that every variable $V \in \mathbf{V}$ is conditionally independent of its non-descendants given its parents, where the descendants of V are all variables reachable by a directed path from V. Deriving the set of conditional independencies from G based on the Markov condition is cumbersome, requiring complex combinations of probability axioms. Fortunately, d-separation, a set of graphical criteria, provides the foundation for algorithmic derivation of all conditional independencies in G and entails the exact same set of conditional independencies as the Markov condition (Verma and Pearl, 1988; Geiger and Pearl, 1988; Neapolitan, 2004).

In the following definition, a path is a sequence of vertices following edges in either direction. We say that a variable V is a collider on a path p if the two arrowheads point at each other (collide) at V; otherwise, V is a non-collider on p.

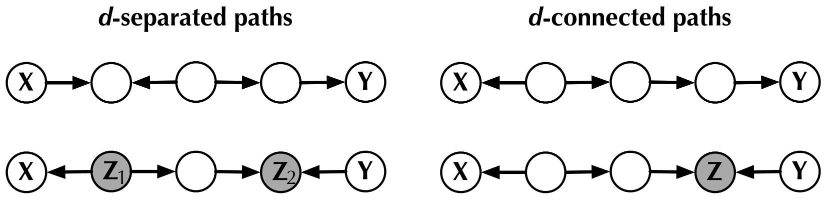
Definition D.1 (*d*-separation) Let $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ be disjoint sets of variables in directed acyclic graph G. A path from some $X \in \mathbf{X}$ to some $Y \in \mathbf{Y}$ is *d-connected* given $\mathbf{Z}$ if and only if every collider W on the path, or a descendant of W, is a member of $\mathbf{Z}$, and there are no non-colliders in $\mathbf{Z}$. Then, say that $\mathbf{X}$ and $\mathbf{Y}$ are *d-separated* by $\mathbf{Z}$ if and only if there are no d-connecting paths between $\mathbf{X}$ and $\mathbf{Y}$ given $\mathbf{Z}$.

Figure D.1(a) depicts the graphical patterns found along paths that lead to d-separation or d-connection based on Definition D.1, and Figure D.1(b) provides example d-separated and d-connected paths. At first glance, identifying conditional independence facts using the rules of d-separation appears computationally intensive, testing a potentially exponential

12. IID data are typically referred to as *propositional* because the data can be equivalently expressed under propositional logic.



(a) Graphical patterns of d-separating and d-connecting path elements among disjoint sets of variables $\mathbf{X}$ and $\mathbf{Y}$ given $\mathbf{Z}$. Paths for which there exists a non-collider in $\mathbf{Z}$ or a collider not in $\mathbf{Z}$ are d-separating. Paths for which all non-colliders are not in $\mathbf{Z}$ and all colliders (or a descendant of colliders) are in $\mathbf{Z}$ are d-connecting.



(b) Several example d-separated and d-connected paths that illustrate the composition of path elements.

Figure D.1: Patterns of d-separating and d-connecting path elements and example d-separating and d-connecting paths.

number of paths. However, Geiger et al. (1990) provide a linear-time algorithm based on breadth-first search and reachability on G.

Under a few assumptions, Bayesian networks can be interpreted causally, with edges corresponding to direct causal dependencies. If $X \rightarrow Y$ is an edge in the causal model G, then manipulating or changing the value of X will alter the conditional distribution of Y—denoted as $P\big(Y \mid do(X)\big)$ using Pearl's *do*-calculus notation for interventions (Pearl, 2000). The causal interpretation of G assumes the *causal* Markov condition, which is identical to the Markov condition, replacing parents with direct causes and non-descendants with non-effects. In order for the causal Markov condition to hold, the variables $\mathbf{V}$ must also be *causally sufficient*: There are no latent common causes for any pair of variables in $\mathbf{V}$. The causal Markov condition is also equivalent to d-separation; therefore, both provide the connection between causal structures and probability distributions.

The conditional independencies entailed by both the causal Markov condition and d-separation hold in all distributions that G represents. A distribution P is *faithful* to G if all conditional independencies in P are entailed by the causal Markov condition on G. If P is assumed to be faithful to G, then there are algorithms that can learn the Markov,

```
SELECT  t1.id, t1.salary, t2.success, t3.revenue
FROM      (  SELECT  E.id, E.salary
             FROM    Employee E) t1,
          (  SELECT  E.id, P.success
             FROM    Employee E, Develops D, Product P
             WHERE   E.id = D.e_id AND D.p_id = P.id) t2,
          (  SELECT  E.id, B.revenue
             FROM    Employee E, Develops D, Product P
                     Funds F, Business-Unit B
             WHERE   E.id = D.e_id AND D.p_id = P.id AND
                     P.id = F.p_id AND F.b_id = B.id) t3
WHERE   t1.id = t2.id AND t2.id = t3.id
```

Figure D.2: Sketch of a relational database query that joins the instances of three relational variables having the common perspective Employee used to produce the data instances shown in Table D.1. The three relational variables are (1) [Employee].*Salary*, (2) [Employee, Develops, Product].*Success*, and (3) [Employee, Develops, Product, Funds, Business-Unit].*Revenue*.

or likelihood, equivalent set of causal models. These algorithms assume causal sufficiency, faithfulness, and model acyclicity to identify the edges in G that are consistent with observed conditional independencies and to determine the direction of causality (Spirtes et al., 2000).

The relational representation presented in Section 4 is strictly more expressive than the propositional representation used in Bayesian network modeling. Propositional representations describe domains with a single entity class; thus, they produce schemas with $|\mathcal{E}| = 1$ (one entity class) and $|\mathcal{R}| = 0$ (no relationship classes). For the organization domain example, consider data about only employees ($\mathcal{E} = \{\text{Employee}\}$). Variables would include intrinsic attributes, such as salary, but could also include variables describing other related entities, all from the employee perspective. This technique of translating a relational database down to a single, propositional representation is often referred to as *propositionalization* (Kramer et al., 2001). That is, we could construct a single table for employees that includes columns for the success of developed products, the revenue of all business units they work under, etc. In Figure D.2, we show an example SQL-like query that would produce such data, and the resulting data set applied to the example in Figure 2.1(b) is shown in Table D.1.[13]

The relational skeleton of a Bayesian network consists of a set of disconnected entity instances, all drawn from the same entity class. Consequently, the skeleton has a simple one-to-one mapping with the representation as a table: Each entity instance corresponds

13. Note that modeling propositionalized data with Bayesian networks still requires the IID assumption, which is often violated since variables of one instance can influence variables of another. For example, according to the model in Figure 2.2(a) the competence of collaborating employees influences the success of products, which affects the revenue of business units, which affects its budget, thereby influencing an employee's salary. As a result, modeling relational data with a propositional representation may unnecessarily lose valuable information, especially in the context of causal reasoning and accurate estimation of causal effects.

Employee	[Employee].*Salary*	[Employee, Develops, Product].*Success*	[Employee, Develops, Product, Funds, Business-Unit].*Revenue*
Paul	{Paul.*Salary*}	{Case.*Success*}	{Accessories.*Revenue*}
Quinn	{Quinn.*Salary*}	{Case.*Success*, Adapter.*Success*, Laptop.*Success*}	{Accessories.*Revenue*, Devices.*Revenue*}
Roger	{Roger.*Salary*}	{Laptop.*Success*}	{Devices.*Revenue*}
Sally	{Sally.*Salary*}	{Laptop.*Success*, Tablet.*Success*}	{Devices.*Revenue*}
Thomas	{Thomas.*Salary*}	{Tablet.*Success*, Smartphone.*Success*}	{Devices.*Revenue*}

Table D.1: Propositional table consisting of employees, their salary, the success of products they develop, and the revenue of the business units they operate under. Producing this table requires joining the instances of three relational variables, all from a common perspective—Employee.

to a single row, and each variable is a column. In this example, each employee would be an entity instance, and no instances of other entity types or relationships would appear in the skeleton. Because all variables in a Bayesian network are defined for a single entity class and no relationships, the relational path specification becomes trivial and, hence, implicit. All relational paths, relational variables, and relational dependencies are defined from a single perspective with singleton paths (e.g., [Employee]). The ground graph of a Bayesian network, similar to the skeleton, has a very regular structure. The ground graph consists of a set of identical copies of the model structure, one for each instance in the skeleton. For a Bayesian network, d-separation can be applied directly to the model structure because there is no variability in its ground graphs.

Appendix E. Hop Thresholds

For practical implementations, the size of the abstract ground graphs should be limited by a domain-specific threshold. In this work, we choose to apply a singular hop threshold to the relational paths that are represented in an abstract ground graph. In this appendix, we examine the effect of choosing a particular hop threshold.

First, we introduce the notion of (B, h)-reachability, which describes the conditions under which an edge in a ground graph is represented in an abstract ground graph.

Definition E.1 ((B, h)-reachability) Let $GG_{\mathcal{M}\sigma}$ be the ground graph for some relational model structure $\mathcal{M}$ and skeleton $\sigma \in \Sigma_{\mathcal{S}}$. Then, $i_k.X \rightarrow i_j.Y \in GG_{\mathcal{M}\sigma}$ is (B, h)*-reachable* for perspective B and hop threshold h if there exist relational variables $P_k.X = [B, \ldots, I_k].X$ and $P_j.Y = [B, \ldots, I_j].Y$ such that $\mathit{length}(P_k) \leq h+1$, $\mathit{length}(P_j) \leq h+1$, and there exists an instance $b \in \sigma(B)$ with $i_k \in P_k|_b$ and $i_j \in P_j|_b$.

In other words, the edge $i_k.X \rightarrow i_j.Y$ in the ground graph is (B, h)-reachable if an instance of the base item $b \in \sigma(B)$ can reach i_k and i_j in at most h hops.

Since Definition E.1 pertains to edges reachable via a particular perspective B and hop threshold h, it relates to the reachability of edges in abstract ground graphs. We denote abstract ground graphs for perspective B, limited by a hop threshold h as $AGG_{\mathcal{M}Bh}$. Definition E.1 implies that (1) for every edge in ground graph $GG_{\mathcal{M}\sigma}$, we can derive a set of abstract ground graphs for which that edge is (B, h)-reachable, and (2) for every abstract ground graph $AGG_{\mathcal{M}Bh}$, we can derive the set of (B, h)-reachable edges for a given ground graph. Given (B, h)-reachability, we can now express the soundness and completeness of abstract ground graphs.

Theorem E.1 *For every acyclic relational model structure $\mathcal{M}$, perspective $B \in \mathcal{E} \cup \mathcal{R}$, and hop threshold $h_a \in \mathbb{N}^0$, the abstract ground graph $AGG_{\mathcal{M}Bh_a}$ is sound up to hop threshold h_a for all ground graphs $GG_{\mathcal{M}\sigma}$ with skeleton $\sigma \in \Sigma_{\mathcal{S}}$.*

Proof. Soundness means that for every edge $[B, \ldots, I_j].X \to [B, \ldots, I_k].Y$ in the abstract ground graph $AGG_{\mathcal{M}Bh_a}$, there exists a skeleton $\sigma \in \Sigma_{\mathcal{S}}$, a base item instance $b \in \sigma(B)$, an instance $i_j \in [B, \ldots, I_j]|b$, and an instance $i_k \in [B, \ldots, I_k]|b$ such that $i_j.X \to i_k.Y$ is a (B, h_a)-reachable edge in $GG_{\mathcal{M}\sigma}$. The proof is identical to the proof of soundness for Theorem 5.2 (see Appendix A). ■

Theorem E.2 *For every acyclic relational model structure $\mathcal{M}$, perspective $B \in \mathcal{E} \cup \mathcal{R}$, and hop threshold $h_r \in \mathbb{N}^0$, the abstract ground graph $AGG_{\mathcal{M}Bh_a}$ is complete up to hop threshold h_r for all ground graphs $GG_{\mathcal{M}\sigma}$ with skeleton $\sigma \in \Sigma_{\mathcal{S}}$, where $h_a = \max(h_r+h_m, h_r+2h_m-2)$ and h_m is the maximum number of hops for a dependency in $\mathcal{M}$.*

Proof. Let $\mathcal{M} = (\mathcal{S}, \mathcal{D})$ be an arbitrary acyclic relational model structure, let $B \in \mathcal{E} \cup \mathcal{R}$ be an arbitrary perspective, and let $h_r \in \mathbb{N}^0$ be an arbitrary hop threshold.

To prove that the abstract ground graph $AGG_{\mathcal{M}Bh_a}$ is complete up to hop threshold h_r, we show that for every (B, h_r)-reachable edge $i_k.X \to i_j.Y$ in every ground graph $GG_{\mathcal{M}\sigma}$ with $\sigma \in \Sigma_{\mathcal{S}}$, there is a set of corresponding edges in $AGG_{\mathcal{M}Bh_a}$. Specifically, the (B, h_r)-reachable edge $i_k.X \to i_j.Y$ yields two sets of relational variables for some $b \in \sigma(B)$, namely $\mathbf{P_k.X} = \{P_k.X \mid i_k \in P_k|_b \;\wedge\; \mathit{length}(P_k) \le h_r + 1\}$ and $\mathbf{P_j.Y} = \{P_j.Y \mid i_j \in P_j|_b \;\wedge\; \mathit{length}(P_j) \le h_r + 1\}$ by Definition E.1. Note that all relational variables in both $\mathbf{P_k.X}$ and $\mathbf{P_j.Y}$ are nodes in $AGG_{\mathcal{M}Bh_a}$. We show that for all $P_k.X \in \mathbf{P_k.X}$ and for all $P_j.Y \in \mathbf{P_j.Y}$ either (a) $P_k.X \to P_j.Y \in AGG_{\mathcal{M}Bh_a}$, (b) $P_k.X \cap P'_k.X \to P_j.Y \in AGG_{\mathcal{M}Bh_a}$ or $P_k.X \cap P'_k.X \to P'_j.Y \in AGG_{\mathcal{M}Bh_a}$, where $i_k \in P'_k|_b$ and $i_j \in P'_j|_b$, or (c) $P_k.X \to P_j.Y \cap P'_j.Y \in AGG_{\mathcal{M}Bh_a}$ or $P'_k.X \to P_j.Y \cap P'_j.Y \in AGG_{\mathcal{M}Bh_a}$, where $i_k \in P'_k|_b$ and $i_j \in P'_j|_b$.

Let $\sigma \in \Sigma_{\mathcal{S}}$ be an arbitrary skeleton, let $i_k.X \to i_j.Y \in GG_{\mathcal{M}\sigma}$ be an arbitrary (B, h_r)-reachable edge drawn from $[I_j, \ldots, I_k].X \to [I_j].Y \in \mathcal{D}$ where $\mathit{length}([I_j, \ldots, I_k]) \le h_m + 1$, and let $P_k.X \in \mathbf{P_k.X}, P_j.Y \in \mathbf{P_j.Y}$ be an arbitrary pair of relational variables. There are three cases:

(a) $P_k \in \mathit{extend}(P_j, [I_j, \ldots, I_k])$. Then, $\mathit{length}(P_k) \le (h_r+1)+(h_m+1)-1 = h_r+h_m+1 \le h_a+1$. Therefore, $P_k.X$ is a node in the abstract ground graph, and $P_k.X \to P_j.Y \in AGG_{\mathcal{M}Bh_a}$ by Definition 5.2.

(b) $P_k \notin \mathit{extend}(P_j, [I_j, \ldots, I_k])$, but $\exists P'_k \in \mathit{extend}(P_j, [I_j, \ldots, I_k])$ such that $i_k \in P'_k|_b$. Then, $\mathit{length}(P'_k) \le (h_r+1)+(h_m+1)-1 = h_r+h_m+1 \le h_a+1$. Therefore, P'_k is a node in the abstract ground graph, $P'_k.X \to P_j.Y \in AGG_{\mathcal{M}Bh_a}$, and $P_k.X \cap P'_k.X \to P_j.Y \in AGG_{\mathcal{M}Bh_a}$ by Definition 5.2.

(c) For all $P_k \in \mathit{extend}(P_j, [I_j, \ldots, I_k])$, it is the case that $i_k \notin P_k.X|_b$. Then by Lemma 5.2, there exists a P'_j such that $i_j \in P'_j|_b$ and there exists a $P''_k \in \mathit{extend}(P'_j, [I_j, \ldots, I_k])$. Given the way P'_j is constructed, its length is bounded by:

$$\mathit{length}(P'_j) \leq \mathit{length}(P_j) + \mathit{length}([I_j, \ldots, I_k]) - 3 \leq (h_r + 1) + (h_m + 1) - 3 = h_r + h_m - 1$$

P''_k intersects with P_k since they both reach i_k, and the length of P''_k is bounded by:

$$\mathit{length}(P''_k) \leq \mathit{length}(P'_j) + \mathit{length}([I_j, \ldots, I_k]) - 1 \leq (h_r + h_m - 1) + (h_m + 1) - 1 = h_r + 2h_m - 1$$

Also by Lemma 5.2, we know that P_j and P'_j intersect. Since $\mathit{length}(P''_k) \leq h_r + 2h_m - 1 \leq h_a + 1$, P''_k is a node in the abstract ground graph, $P''_k.X \to P'_j.Y \in AGG_{\mathcal{M}Bh_a}$ $P''_k.X \to P'_j.Y \cap P_j.Y \in AGG_{\mathcal{M}Bh_a}$, and $P_k.X \cap P''_k.X \to P'_j.Y \in AGG_{\mathcal{M}Bh_a}$ by Definition 5.2.

From the above three cases, it follows that to guarantee completeness up to h_r, the abstract ground graph must contain nodes up to the hop threshold $h_a = \max(h_r + h_m, h_r + 2h_m - 2)$. ■

Theorems E.1 and E.2 guarantee that if an abstract ground graph is constructed with a hop threshold of h_a from perspective B, it captures all paths of dependence in all ground graphs, where (1) the variables along those paths are reachable in h_r hops from instances of B and (2) the underlying dependencies are bounded by a threshold of h_m.

In the following, we say that d-separation holds up to a specified hop threshold h if there are no d-connecting paths involving relational variables of length greater than $h + 1$.

Theorem E.3 *Relational d-separation is sound and complete for abstract ground graphs up to a specified hop threshold. Let $\mathcal{M}$ be an acyclic relational model structure, and let h_m be the maximum number of hops for a dependency in $\mathcal{M}$. Let $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$ be three distinct sets of relational variables for perspective $B \in \mathcal{E} \cup \mathcal{R}$ defined over relational schema $\mathcal{S}$, and let h_r be the maximum number of hops of relational variables in $\mathbf{X}, \mathbf{Y}$, and $\mathbf{Z}$. Then, $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ are d-separated by $\bar{\mathbf{Z}}$ on the abstract ground graph $AGG_{\mathcal{M}Bh_a}$ if and only if for all skeletons $\sigma \in \Sigma_{\mathcal{S}}$ and for all $b \in \sigma(B)$, $\mathbf{X}|_b$ and $\mathbf{Y}|_b$ are d-separated by $\mathbf{Z}|_b$ up to hop threshold h_r in ground graph $GG_{\mathcal{M}\sigma}$, where $h_a = \max(h_r + h_m, h_r + 2h_m - 2)$.*

Proof. We must show that d-separation on an abstract ground graph implies d-separation on all ground graphs it represents (soundness) and that d-separation facts that hold across all ground graphs are also entailed by d-separation on the abstract ground graph (completeness).

Soundness: Assume that $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ are d-separated by $\bar{\mathbf{Z}}$ on $AGG_{\mathcal{M}Bh_a}$. Assume for contradiction that there exists a skeleton $\sigma \in \Sigma_{\mathcal{S}}$ and an item instance $b \in \sigma(B)$ such that $\mathbf{X}|_b$ and $\mathbf{Y}|_b$ are *not* d-separated by $\mathbf{Z}|_b$ in the ground graph $GG_{\mathcal{M}\sigma}$. Then, there must exist a d-connecting path p from some $x \in \mathbf{X}|_b$ to some $y \in \mathbf{Y}|_b$ given all $z \in \mathbf{Z}|_b$ such that every edge of p is (B, h_r)-reachable. By Theorem E.2, $AGG_{\mathcal{M}Bh_a}$ is (B, h_r)-reachably complete, so all (B, h_r)-reachable edges in $GG_{\mathcal{M}\sigma}$ are captured by edges in $AGG_{\mathcal{M}Bh_a}$. Thus, path p must be represented from some node in $\{N_x \mid x \in N_x|_b\}$ to some node in $\{N_y \mid y \in N_y|_b\}$, where N_x, N_y are nodes in $AGG_{\mathcal{M}Bh_a}$. If p is d-connecting in $GG_{\mathcal{M}\sigma}$,

Predictor	Coefficient	Partial	Semipartial
log(# dependencies) $\times$ # entities	1.38	0.232	0.085
log(# dependencies)	1.14	0.135	0.044
log(# dependencies) $\times$ # MANY cardinalities	-0.71	0.092	0.028
# entities $\times$ # relational variables	-0.32	0.044	0.013

Table F.1: Number of equivalent conditional independence judgments: estimated standardized coefficient, squared partial correlation coefficient, and squared semipartial correlation coefficient for each predictor.

then it is d-connecting in $AGG_{\mathcal{M}Bh_a}$, which implies that $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ are *not* d-separated by $\bar{\mathbf{Z}}$. Therefore, $\mathbf{X}|_b$ and $\mathbf{Y}|_b$ must be d-separated by $\mathbf{Z}|_b$.

Completeness: Assume that $\mathbf{X}|_b$ and $\mathbf{Y}|_b$ are d-separated by $\mathbf{Z}|_b$ in the ground graph $GG_{\mathcal{M}\sigma}$ for all skeletons $\sigma \in \Sigma_{\mathcal{S}}$ and for all $b \in \sigma(B)$. Assume for contradiction that $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ are *not* d-separated by $\bar{\mathbf{Z}}$ on $AGG_{\mathcal{M}Bh_a}$. Then, there must exist a d-connecting path p for some relational variable $X \in \bar{\mathbf{X}}$ to some $Y \in \bar{\mathbf{Y}}$ given all $Z \in \bar{\mathbf{Z}}$. By Theorem E.1, $AGG_{\mathcal{M}Bh_a}$ is (B, h_a)-reachably sound, so every edge in $AGG_{\mathcal{M}Bh}$ must correspond to some pair of variables in some ground graph. Thus, if p is d-connecting in $AGG_{\mathcal{M}Bh_a}$, then there must exist some skeleton σ such that p is d-connecting in $GG_{\mathcal{M}\sigma}$ for some $b \in \sigma(B)$, which implies that d-separation does not hold for that ground graph. Therefore, $\bar{\mathbf{X}}$ and $\bar{\mathbf{Y}}$ must be d-separated by $\bar{\mathbf{Z}}$ on $AGG_{\mathcal{M}Bh_a}$. ∎

Appendix F. Experimental Details—Equivalence of a Naïve Approach

In this appendix, we provide additional details for the experiment in Section 6. The main goal of this experiment is to quantify how often traditional d-separation applied directly to relational model structures produces incorrect conditional independence facts. This provides a rough measurement for the additional representational power of relational d-separation on abstract ground graphs. Here, we present an analysis of which factors influence the number of equivalent and non-equivalent conditional independence judgments between both approaches (naïvely applying traditional d-separation versus relational d-separation).

Specifically, we show here the results of running log-linear regression to predict the number of equivalent and non-equivalent judgments for varying schemas and models. We first applied lasso for feature selection (Tibshirani, 1996) to minimize the number of predictors while maximizing model fit. We also standardized the input variables by dividing by two standard deviations, as recommended by Gelman (2008). Since the predictor for the number of dependencies is log-transformed, the standardization for that variable occurs after taking the logarithm.

In predicting the (log of the) number of equivalent conditional independencies, the following variables were significantly and substantively predictive (in order of decreasing predictive power):

- Interaction between the log of the number of dependencies and the number of entities (positive)
- Log of the number of dependencies (positive)

Predictor	Coefficient	Partial	Semipartial
# MANY cardinalities × # entities	-2.22	0.207	0.064
log(# dependencies) × # entities	0.90	0.165	0.048
# MANY cardinalities	3.24	0.128	0.036
log(# dependencies) × # MANY cardinalities	1.47	0.127	0.036

Table F.2: Number of non-equivalent conditional independence judgments: estimated standardized coefficient, squared partial correlation coefficient, and squared semipartial correlation coefficient for each predictor.

- Interaction between the log of the number of dependencies and the number of MANY cardinalities (negative)
- Number of entities (negative)
- Interaction between the number of entities and the number of relational variables in the AGG (negative)

The fit for the equivalent model has an $R^2 = 0.721$ for $n = 4,000$, and Table F.1 contains the standardized coefficients as well as the squared partial and semipartial correlation coefficients for each predictor. For lasso, $\lambda = 0.0076$ offered the fewest predictors while increasing the model fit by at least 0.01.

In predicting the (log of the) number of non-equivalent conditional independencies, the following variables were significantly and substantively predictive (in order of decreasing predictive power):

- Interaction between the number of MANY cardinalities and the number of entities (negative)
- Interaction between the log of the number of dependencies and the number of entities (positive)
- Number of MANY cardinalities (positive)
- Interaction between the log of the number of dependencies and the number of MANY cardinalities (positive)

The fit for the non-equivalent model has an $R^2 = 0.755$ for $n = 4,000$, and Table F.2 contains the standardized coefficients and the squared partial and semipartial correlation coefficients for each predictor. For lasso, $\lambda = 0.0155$ offered the fewest predictors while increasing the model fit by at least 0.01.

Appendix G. Experimental Details—Abstract Ground Graph Size

In this appendix, we provide additional details for the experiment in Section 7.1. The goal of this experiment is to determine which factors influence the size of abstract ground graphs because the computational complexity of relational d-separation depends on their size. Specifically, we show here the results of running log-linear regression to predict the size of abstract ground graphs for varying schemas and models. We first applied lasso for feature selection (Tibshirani, 1996) to minimize the number of predictors while maximizing

Predictor	Coefficient	Partial	Semipartial
# relationships	3.24	0.452	0.150
# MANY cardinalities × isEntity=F	3.09	0.349	0.109
# entities	-2.11	0.359	0.102
# MANY cardinalities × isEntity=T	2.51	0.216	0.053
# MANY cardinalities × # relationships	-0.88	0.100	0.020
# attributes	0.23	0.024	0.004

Table G.1: Number of nodes in an abstract ground graph: estimated standardized coefficient, squared partial correlation coefficient, and squared semipartial correlation coefficient for each predictor.

Predictor	Coefficient	Partial	Semipartial
log(# dependencies)	1.44	0.440	0.165
# relationships	3.86	0.395	0.138
# MANY cardinalities × isEntity=F	4.27	0.356	0.123
# entities	-2.78	0.353	0.115
# MANY cardinalities × isEntity=T	3.52	0.231	0.067
# MANY cardinalities × # relationships	-1.35	0.127	0.031

Table G.2: Number of edges in an abstract ground graph: estimated standardized coefficient, squared partial correlation coefficient, and squared semipartial correlation coefficient for each predictor.

model fit. We also standardized the input variables by dividing by two standard deviations, as recommended by Gelman (2008). Since the predictor for the number of dependencies is log-transformed, the standardization for that variable occurs after taking the logarithm.

In predicting the (log of the) number of nodes, the following variables were significantly and substantively predictive (in order of decreasing predictive power):

- Number of relationships (positive)
- Interaction between MANY cardinalities and an indicator variable for whether the abstract ground graph is from an entity or relationship perspective (positive)
- Number of entities (negative)
- Interaction between the number of MANY cardinalities and relationships (negative)
- Total number of attributes (positive)

The fit for the nodes model has an $R^2 = 0.818$ for $n = 450,000$, and Table G.1 contains the standardized coefficients as well as the squared partial and semipartial correlation coefficients for each predictor. For lasso, $\lambda = 0.0095$ offered the fewest predictors while increasing the model fit by at least 0.01.

In predicting the (log of the) number of edges, the following variables were significantly and substantively predictive (in order of decreasing predictive power):

- Log of the number of dependencies (positive)

- Number of relationships (positive)
- Interaction between MANY cardinalities and an indicator variable for whether the abstract ground graph is from an entity or relationship perspective (positive)
- Number of entities (negative)
- Interaction between the number of MANY cardinalities and relationships (negative)

The fit for the edges model has an $R^2 = 0.789$ for $n = 450,000$, and Table G.2 contains the standardized coefficients and the squared partial and semipartial correlation coefficients for each predictor. For lasso, $\lambda = 0.0164$ offered the fewest predictors while increasing the model fit by at least 0.01.

References

Richard Barker. *CASE Method: Entity Relationship Modeling.* Addison-Wesley, Boston, MA, 1990.

Christopher M. Bishop. Latent variable models. In Michael I. Jordan, editor, *Learning in Graphical Models*, pages 371–403. MIT Press, Cambridge, MA, 1999.

Craig Boutilier, Nir Friedman, Moises Goldszmidt, and Daphne Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 115–123, 1996.

Wray L. Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

Jie Cheng, David A. Bell, and Weiru Liu. Learning belief networks from data: An information theory based approach. In *Proceedings of the Sixth International Conference on Information and Knowledge Management*, pages 325–331, 1997.

Tom Claassen and Tom Heskes. A logical characterization of constraint-based causal discovery. In *Proceedings of Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pages 135–144, 2011.

Tom Claassen and Tom Heskes. A Bayesian approach to constraint based causal inference. In *Proceedings of Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 207–216, 2012.

Diego Colombo, Marloes H. Maathuis, Markus Kalisch, and Thomas S. Richardson. Learning high-dimensional directed acyclic graphs with latent and selection variables. In *The Annals of Statistics*, volume 40, pages 294–321, 2012.

Bruce D'Ambrosio, Eric Altendorf, and Jane Jorgensen. Ecosystem analysis using probabilistic relational modeling. In *Proceedings of the International Joint Conference on Artificial Intelligence Workshop on Learning Statistical Models from Relational Data*, 2003.

Denver Dash. Restructuring dynamic causal systems in equilibrium. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 81–88, 2005.

Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

Peter A. Flach. Knowledge representation for inductive learning. In *Proceedings of the Fifth European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, pages 160–167, 1999.

Nir Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, 303 (5659):799–805, 2004.

Nir Friedman, Lise Getoor, Daphne Koller, and Avi Pfeffer. Learning probabilistic relational models. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, volume 16, pages 1300–1309, 1999.

Maxime Gasse, Alex Aussem, and Haytham Elghazel. An experimental comparison of hybrid algorithms for Bayesian network structure learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 58–73, 2012.

Dan Geiger and Judea Pearl. On the logic of causal models. In *Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*, pages 136–147, 1988.

Dan Geiger, Thomas Verma, and Judea Pearl. Identifying independence in Bayesian networks. *Networks*, 20(5):507–534, 1990.

Andrew Gelman. Scaling regression inputs by dividing by two standard deviations. *Statistics in Medicine*, 27(15):2865–2873, 2008.

Andrew Gelman and Jennifer Hill. *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press New York, 2007.

Lise Getoor. *Learning Statistical Models from Relational Data*. Ph.D. thesis, Stanford University, 2001.

Lise Getoor and Ben Taskar, editors. *Introduction to Statistical Relational Learning*. MIT Press, Cambridge, MA, 2007.

Lise Getoor, Daphne Koller, and Nir Friedman. From instances to classes in probabilistic relational models. In *Proceedings of the International Conference on Machine Learning Workshop on Attribute-Value and Relational Learning*, 2000.

Lise Getoor, Nir Friedman, Daphne Koller, and Ben Taskar. Learning probabilistic models of link structure. 3:679–707, 2002.

Lise Getoor, Jeanne T. Rhee, Daphne Koller, and Peter Small. Understanding tuberculosis epidemiology using structured statistical models. *Artificial Intelligence in Medicine*, 30 (3):233–256, 2004.

Lise Getoor, Nir Friedman, Daphne Koller, Avi Pfeffer, and Ben Taskar. Probabilistic relational models. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, chapter 5, pages 129–174. MIT Press, Cambridge, MA, 2007.

Walter R. Gilks, Andrew Thomas, and David J. Spiegelhalter. A language and program for complex Bayesian modeling. *The Statistician*, 43:169–177, 1994.

David Heckerman, Christopher Meek, and Daphne Koller. *Probabilistic Models for Relational Data.* Technical Report MSR-TR-2004-30, Microsoft Research, Redmond, WA, March 2004.

David Heckerman, Chris Meek, and Daphne Koller. Probabilistic entity-relationship models, PRMs, and plate models. In Lise Getoor and Ben Taskar, editors, *Introduction to Statistical Relational Learning*, chapter 7, pages 201–238. MIT Press, Cambridge, MA, 2007.

Yimin Huang and Marco Valtorta. Identifiability in causal Bayesian networks: A sound and complete algorithm. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 1149–1154, 2006.

Michael G. Hudgens and M. Elizabeth Halloran. Toward causal inference with interference. *Journal of the American Statistical Association*, 103(482):832–842, 2008.

Antti Hyttinen, Frederick Eberhardt, and Patrik O. Hoyer. Learning linear cyclic causal models with latent variables. *Journal of Machine Learning Research*, 13:3387–3439, 2012.

David Jensen and Jennifer Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 259–266, 2002.

Kristian Kersting and Luc De Raedt. *Basic Principles of Learning Bayesian Logic Programs.* Technical Report 174, Institute for Computer Science, University of Freiberg, 2002.

Daphne Koller and Avi Pfeffer. Probabilistic frame-based systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pages 580–587, 1998.

Stefan Kramer, Nada Lavrač, and Peter Flach. Propositionalization approaches to relational data mining. In Sašo Džeroski and Nada Lavrač, editors, *Relational Data Mining*, pages 262–286. Springer-Verlag, New York, NY, 2001.

Kathryn B. Laskey. MEBN: A language for first-order Bayesian knowledge bases. *Artificial Intelligence*, 172(2):140–178, 2008.

Marc Maier, Brian Taylor, Hüseyin Oktay, and David Jensen. Learning causal models of relational domains. In *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence*, pages 531–538, 2010.

Marc Maier, Katerina Marazopoulou, David Arbour, and David Jensen. A sound and complete algorithm for learning causal models from relational data. In *Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence*, pages 371–380, 2013.

Dimitris Margaritis and Sebastian Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems 12*, pages 505–511, 1999.

Brian Milch, Bhaskara Marthi, Stuart J. Russell, David Sontag, Daniel L. Ong, and Andrey Kolobov. BLOG: Probabilistic models with unknown objects. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1352–1359, 2005.

Tom Minka and John Winn. Gates. In *Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems*, pages 1073–1080, 2009.

Kevin P. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning.* Ph.D. thesis, University of California, Berkeley, 2002.

Richard E. Neapolitan. *Learning Bayesian Networks.* Pearson Prentice Hall, Upper Saddle River, NJ, 2004.

Jennifer Neville and David D. Jensen. Leveraging relational autocorrelation with latent group models. In *Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 322–329, 2005.

Jennifer Neville and David D. Jensen. Relational dependency networks. *Journal of Machine Learning Research*, 8:653–692, May 2007.

Judea Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge University Press, New York, NY, 2000.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Fransico, CA, 1988.

Judea Pearl and Rina Dechter. Identifying independencies in causal graphs with feedback. In *Proceedings of the Twelfth International Conference on Uncertainty in Artificial Intelligence*, pages 420–426, 1996.

Judea Pearl and Thomas S. Verma. A theory of inferred causation. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*, pages 441–452, 1991.

Jean-Philippe Pellet and André Elisseeff. Using Markov blankets for causal structure learning. *Journal of Machine Learning Research*, 9:1295–1342, 2008.

Claudia Perlich and Foster Provost. Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning*, 62(1-2):65–105, 2006.

David Poole. First-order probabilistic inference. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, volume 3, pages 985–991, 2003.

David Poole. Logical generative models for probabilistic reasoning about existence, roles and identity. In *Proceedings of the Twenty-Second National Conference on Artificial Intelligence*, pages 1271–1277, 2007.

Raghu Ramakrishnan and Johannes Gehrke. *Database Management Systems.* McGraw-Hill, Inc., New York, NY, 2nd edition, 2002.

Matthew Richardson and Pedro Domingos. Markov logic networks. *Machine Learning*, 62 (1–2):107–136, 2006.

Thomas Richardson and Peter Spirtes. Ancestral graph Markov models. *The Annals of Statistics*, 30(4):962–1030, 2002.

Thomas S. Richardson. *Feedback Models: Interpretation and Discovery.* Ph.D. thesis, Carnegie Mellon University, 1996.

Thomas S. Richardson. A factorization criterion for acyclic directed mixed graphs. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 462–470, 2009.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 3rd edition, 2010.

Richard Scheines. An introduction to causal inference. In Vaughan R. McKim and Steven P. Turner, editors, *Causality in Crisis? Statistical Methods and the Search for Causal Knowledge in the Social Sciences*, pages 185–199. University of Notre Dame Press, 1997.

Mark Schmidt and Kevin Murphy. Modeling discrete interventional data using directed cyclic graphical models. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 487–495, 2009.

Oliver Schulte, Hassan Khosravi, and Tong Man. Learning directed relational models with recursive dependencies. *Machine Learning*, 89(3):299–316, 2012.

Eran Segal, Ben Taskar, Audrey Gasch, Nir Friedman, and Daphne Koller. Rich probabilistic models for gene expression. *Bioinformatics*, 17(suppl 1):S243–S252, 2001.

Cosma R. Shalizi and Andrew C. Thomas. Homophily and contagion are generically confounded in observational social network studies. *Sociological Methods & Research*, 40(2): 211–239, 2011.

Chuan Shi, Xiangnan Kong, Philip S. Yu, Sihong Xie, and Bin Wu. Relevance search in heterogeneous networks. In *Proceedings of the Fifteenth International Conference on Extending Database Technology*, pages 180–191, 2012.

Ilya Shpitser and Judea Pearl. Complete identification methods for the causal hierarchy. *Journal of Machine Learning Research*, 9:1941–1979, 2008.

Teodor Sommestad, Mathias Ekstedt, and Pontus Johnson. A probabilistic relational model for security risk analysis. *Computers & Security*, 29(6):659–679, 2010.

Peter Spirtes. Directed cyclic graphical representations of feedback models. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 491–498, 1995.

Peter Spirtes, Christopher Meek, and Thomas Richardson. Causal inference in the presence of latent variables and selection bias. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 499–506, 1995.

Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction and Search.* MIT Press, Cambridge, MA, 2nd edition, 2000.

Yizhou Sun, Jiawei Han, Xifeng Yan, Philip S. Yu, and Tianyi Wu. PathSim: Meta path-based top-k similarity search in heterogeneous information networks. In *Proceedings of the VLDB Endowment*, pages 992–1003, 2011.

Ben Taskar, Eran Segal, and Daphne Koller. Probabilistic classification and clustering in relational data. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 870–878, 2001.

Ben Taskar, Pieter Abbeel, and Daphne Koller. Discriminative probabilistic models for relational data. In *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, pages 485–492, 2002.

Eric J. Tchetgen Tchetgen and Tyler J. VanderWeele. On causal inference in the presence of interference. *Statistical Methods in Medical Research*, 21(1):55–75, 2012.

Jin Tian and Judea Pearl. A general identification condition for causal effects. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence*, pages 567–573, 2002.

Jin Tian, Azaria Paz, and Judea Pearl. *Finding Minimal D-separators.* Technical Report R-254, UCLA Computer Science Department, February 1998.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78, October 2006.

Thomas Verma and Judea Pearl. Causal networks: Semantics and expressiveness. In *Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*, pages 352–359, 1988.

John Winn. Causality with gates. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, pages 1314–1322, 2012.

Rongjing Xiang and Jennifer Neville. Relational learning with one network: An asymptotic analysis. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 779–788, 2011.

Raanan Yehezkel and Boaz Lerner. Bayesian network structure learning by recursive autonomy identification. *Journal of Machine Learning Research*, 10:1527–1570, 2009.

Jiji Zhang. On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. *Artificial Intelligence*, 172(16):1873–1896, 2008.